A Self-Adaptive and Secure Approach to Share Network Trace Data

ANTONIOS XENAKIS, University of Maryland Baltimore County, USA
SABRINA MAMTAZ NOURIN, University of Maryland Baltimore County, USA
ZHIYUAN CHEN, University of Maryland Baltimore County, USA
GEORGE KARABATIS, University of Maryland Baltimore County, USA
AHMED ALEROUD, Augusta University, USA
JHANCY AMARSINGH, University of Maryland Baltimore County, USA

A large volume of network trace data are collected by the government, public, and private organizations, and can be analyzed for various purposes such as resolving network problems, improving network performance, and understanding user behavior. However, most organizations are reluctant to share their data with any external experts for analysis because it contains sensitive information deemed proprietary to the organization, thus raising privacy concerns. Even if the payload of network packets is not shared, header data may disclose sensitive information that adversaries can exploit to perform unauthorized actions. So network trace data needs to be anonymized before being shared. Most of existing anonymization tools have two major shortcomings: 1) they cannot provide provable protection; 2) their performance relies on setting the right parameter values such as the degree of privacy protection and the features that should be anonymized, but there is little assistance for a user to optimally set these parameters. This paper proposes a self-adaptive and secure approach to anonymize network trace data, and provides provable protection and automatic optimal settings of parameters. A comparison of the proposed approach with existing anonymization tools via experimentation demonstrated that the proposed method outperforms the existing anonymization techniques.

CCS Concepts: • **Security and privacy → Network security**.

Additional Key Words and Phrases: privacy protection, anonymization

ACM Reference Format:
Antonios Xenakis, Sabrina Mamtaz Nourin, Zhiyuan Chen, George Karabatis, Ahmed Aleroud, and Jhancy Amarsingh. 2023. A Self-Adaptive and Secure Approach to Share Network Trace Data. In . ACM, New York, NY, USA, 21 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Many organizations in the government, public, or private sector generate a large volume of network traces, consisting of data packets sent through a computer network. The information contained within these traces can be analyzed, and the results of such analyses can be utilized to achieve organizational goals, such as to maintain network health, improve network performance, or understand user behavior. However, most organizations prefer not to share their internal data with any external entities for analysis because it contains sensitive and proprietary information about the organization, thus raising privacy concerns. Some attempts to solve this problem include the removal of payload data, but even if the payload is removed and not shared, header data may disclose sensitive information that adversaries

Manuscript submitted to ACM

can exploit to perform unauthorized actions. For example, IP addresses can be exploited by adversaries to identify hosts in a network. Port numbers may be used to identify applications that generate the trace. This information can be used in fingerprinting attacks to reveal that a certain application with suspected vulnerabilities is running on a network. On the other hand, organizations must comply with privacy restrictions [27]. The sharing of data among different Federal agencies or departments in the US government is codified and mandatory in an information sharing environment through the use of anonymized data. Furthermore, "the use of such information is reasonably expected to produce results materially equivalent to the use of information that is transferred or stored in a non-anonymized form" and "such use is consistent with any mission of that department, agency, or component (including any mission under a Federal statute or directive of the President) that involves the storage, retention, sharing, or exchange of personally identifiable information."[26]. Therefore, data anonymization prior to sharing becomes a necessary, convenient and sometimes required solution to organizations.

Anonymizing network data is not a simple task. On the contrary, it is quite a challenging one, as there are many domain-specific requirements and constraints that one must balance for successful anonymization. In general, an efficient and cost-effective anonymization method has to balance between the risk of re-identifying the original data, and maintaining the utility of the anonymized data. Although, several anonymization methods are available [3, 21], they have two major shortcomings. First, most of these methods cannot provide provable protection. Many of the earlier methods rely on just anonymizing IP addresses [28]. However, as shown in [6] adversaries can inject packets with specific patterns to be included in the dataset that will be anonymized, with the purpose of identifying these injected patterns after the anonymization process, and subsequently re-identifying records, and potentially revealing additional information about the original data. Second, the performance of most existing anonymization methods relies on setting the appropriate parameter values, such as the degree of privacy protection and the features that should be anonymized, but there is little assistance for a user to set these parameters optimally. For example, it is quite important to identify and select the best combination of features to anonymize in order to optimize both the usefulness of the anonymized data as well as the degree of privacy protection. Such decisions often depend on the type of analysis to be performed on the anonymized data as well as the statistical properties of the data. A feature that is very important for the analysis probably should not be anonymized. On the other hand, a feature that has many rare values probably should be anonymized because it could be used by adversaries to reveal the identity of a record. This paper contains detailed descriptions of the following contributions:

(1) A secure technique to anonymize network trace data. This approach provides a provable protection based on differential privacy [10].
(2) A novel method to select features to anonymize based on properties of the data.
(3) A new approach to automatically identify optimal settings of parameters for the proposed anonymization technique.

The reminder of the paper is organized as follows. Section 2 describes related work. Section 3 proposes our solution. Section 4 presents experimental results. Section 5 concludes the paper.

2 RELATED WORK

Related work on privacy protection techniques for general data can be mostly divided into two types of approaches: 1) masking or obscuring the values of fields that may be used to identify an individual or a device/machine (e.g., K-anonymity [25]) and 2) maintaining privacy of individuals by adding noise to results of a query (e.g., differential

privacy [10]). The first approach better preserves data properties useful for subsequent analysis but does not provide strong privacy protection. For example, adversaries can inject packets with specific patterns into network data and even when the IP addresses are encrypted, adversaries can still re-identify injected packets and then use them to reveal the binding between the original and anonymized data [6]. The second approach provides strong privacy protection, but it often adds too much noise such that useful data properties are not preserved; consequently, most anonymization methods for network data use the first approach.

For example, PCAPLIB [18] utilizes Wireshark dissectors to parse many of the protocol features and can therefore anonymize up to the application layer. It uses an anonymization approach that preserves the semantics and length of the fields. Crypto-PAn [28] anonymizes IP addresses using a prefix preserving anonymization approach, but it ignores anonymization of other features. A more efficient online prefix-preserving method was proposed in [7]. A more recent work uses the Feistel Network encryption method on IP addresses to preserve prefixes [8] and allows decryption. TCPANON [14] is another tool that uses customized parsing to hide identities. Other tools such as SCRUB-TCPDUMP [17] anonymize a small number of traffic features so their functionality is limited. Finally, tools such as TRACEWRANGLER [4] only sanitize network captures. Note that all these methods do not provide provable privacy protection such as differential privacy. Anonymized data can be still a target of several attacks, which include injection or inspection attacks[6]. Most attacks that target anonymized flows are passive and can be executed after anonymization. Such attacks work by matching the features of anonymized flows with the features of original flows to break anonymization functions.

Data injection attacks are initiated by injecting records with known patterns in the original data sets then trying to re-identify these records after anonymization. It is similar to the chosen plain text attack that targets encryption algorithms. An injection attack has been described as a privileged attack since it gives an attacker more insight about the data set compared to an external observer [6]. Data injection attacks are usually executed by sending IP packets with random source and destination IP addresses, sending packets by spoofing IP addresses or sending packets by forging packet headers to appear as anonymized traffic [5]. The adversary can then discover the mapping between the injected packets before and after anonymization. Such attacks also succeed if the adversaries have some knowledge of the original flows from relevant sources such as DNS and WHOIS. They may then inject some fake flows to the original traces; then it then becomes easy to discover those fake flows in the anonymized data using matching algorithms. The adversaries may also escalate their fingerprinting activities to discover other sensitive attributes such as the shared prefixes of IP addresses in the network. Mohammady et al. mentioned that many of the existing anonymization schemes have been shown to be vulnerable to injection attacks [21], including some very well-known anonymization approaches such as Crypto-PAn [28]. In this paper we take into consideration injection attacks that inject records with known patterns such as those in [6]. Mohammady et. al. [21] proposed a solution to make prefix-preserving encryption robust to injection attacks, but their solution requires creating multiple variants of the same data and the recipient of the anonymized data must run analysis multiple times (once for each variant). On the contrary, our approach does not require any additional work.

Only two works [3, 19] we are aware of use differential privacy for network trace data. However, the work by McSherry and Mahajan does not allow data recipients to have direct access to anonymized data, instead they have to use a query interface provided by data curator [19]. This often does not work in practice as data recipients typically prefer to use their own analysis methods/tools. Aleroud et al. [3] propose a condensation-based solution, but the clustering method used does not limit the impact of a single record (adding a record may cause major changes to many clusters), and does not guarantee differential privacy. Another shortcoming of existing work is the requirements that users

should select appropriate features of the data set to anonymize, which is often not a trivial task for users. A survey on anonymizing network data [9] shows that a major difference between anonymizing network data versus other types of data is the existence of domain specific requirements that dictate how network data will be used in analysis. For example, it is often necessary to preserve certain parts of a field e.g., the prefix of an IP or MAC address. On the other hand, the re-identification risk often depends on statistical characteristics of the data. For example, within a specific data set, the vendor code of a MAC address may be unique, hence, it carries a high re-identification risk, while in another data set there could be many MAC addresses with the same vendor code resulting in a much lower re-identification risk. This paper proposes a solution that selects features to anonymize and considers both subsequent analysis and re-identification risks.

There has been a rich body of work on hyperparameter tuning in machine learning [29]. However, there is little work on parameter tuning for data anonymization techniques. This paper proposes a solution, which also includes a method tailored to the specific problem of automated parameter tuning for anonymization purposes.

3 PROPOSED METHODOLOGY

This section provides a novel method to anonymize a network trace. It supports both differential privacy [10] and k-anonymity [25]. k-anonymity is an easy to understand privacy model and it prevents attacks that link an anonymized data set with other public data sets to reveal a person's identity. It usually leads to better data utility as well. Differential privacy is a stronger privacy model with a provable guarantee.

Differential privacy allows a user to specify the exact point between the degree of privacy and the level of accuracy of a data set. This point is represented by the privacy-loss budget or simply the privacy budget. The privacy budget ϵ ranges from 0 (full privacy but no accuracy) to ∞ (no privacy but absolute accuracy). It is used in the differential privacy model to provide provable privacy protection. A larger budget allows more accurate analysis results on anonymized data but provides less privacy protection. For example, in the US 2020 Census a high privacy budget of 19.61 was used [16] resulting in more accurate results.

More specifically, let D_1 and D_2 be two data sets that differ by just one record. Let D' be an anonymized data set. An anonymization method $\mathcal{A}$ is differentially private if the probability of generating D' from D_1 to the probability of getting D' from D_2 is bounded by $e^{\pm\epsilon}$. Intuitively, differential privacy prevents adversaries from guessing whether a record belongs to D based on D'. Formally, an anonymization method $\mathcal{A}$ is differentially private if it is bounded as shown in Equation 3.1.

$$e^{-\epsilon} \leq \frac{P(\mathcal{A}(D_1) = D')}{P(\mathcal{A}(D_2) = D')} \leq e^{\epsilon} \tag{3.1}$$

where ϵ is the privacy budget. Usually the smaller the ϵ, the better the privacy protection and the lower the accuracy due to increased level of added noise [11].

Regarding k-anonymity, the value of k requires that each record has at least $k-1$ other records that have the same value on anonymized features. Consequently, adversaries cannot uniquely identify a record based on these features. However, k-anonymity does not provide a worst-case privacy guarantee as adversaries with additional knowledge may still identify a record. This is not a problem in our method since our solution also supports differential privacy, which does provide a worst-case privacy guarantee.

Both differential privacy and k-anonymity require values for their input parameters, the privacy budget ϵ, and the cluster size k respectively. Those values must be known before anonymization takes place and must be set accordingly in order to optimize the trade-off between accuracy and privacy protection.

Algorithm 1 describes our overall anonymization algorithm for network traces. The core of this algorithm is a clustering method that divides data into small clusters and replaces individual instances in each cluster with an aggregated instance. To support differential privacy, the clustering is not sensitive to the addition or deletion of a single instance, i.e., only a limited number of instances will change their cluster membership when a new record is added. Such clustering method is called Insensitive MicroAggregation.

Hence, we call our method Anonymization with Insensitive MicroAggregation. It provides strong privacy protection because it supports both k-anonymity and differential privacy.

The input to this algorithm is a data set D, a privacy budget ϵ, and a k-anonymity measure k. Algorithm 1 first identifies IP or MAC addresses in the data set and applies prefix-preserving anonymization on these features. Prefixes of such features often indicate whether two nodes belong to the same subnet and the remaining digits represent the specific node within a subnet. In cyber security applications, it is important to ensure that if two nodes belong to the same subnet, their anonymized IP addresses still belong to the same subnet, therefore, the anonymized instances of such nodes should reflect this property.

Algorithm 1 Anonymization with Insensitive MicroAggregation

procedure ANONYMIZE_WITH_IMA(D, ϵ, k)
  Prefix preserving anonymization of IP or MAC address features
  Select other features to anonymize based on data properties
  Divide D into subsets $D_1, \ldots, D_l$ such that each D_i contains records with class label y_i. If there is no class label, just use one subset $D_1 = D$.
  for Each D_i **do**
    Run insensitive microaggregation to divide D_i into clusters $C_{i1}, \ldots, C_{in}$
    for Each cluster $C_{ij}, 1 \leq j \leq n$ **do**
      for numerical features, compute mean of C_{ij} on features to be anonymized and use Laplace mechanism [10] to add noise to these features
      for categorical features, use the exponential mechanism [20] to compute the perturbed most frequent value within the cluster.
      Replace each record in C_{ij} with the perturbed mean or most frequent value.
    end for
  end for
end procedure

Algorithm 1 next applies a method to select additional features from D to anonymize, as network trace data often contain many features. Note that anonymizing all features in D often damages important properties of the data that are important for subsequent analysis. For example, a port number often indicates the type of services provided at the host so anonymizing it may cause issues in subsequent analysis. On the other hand, some features may have many unique values in the data set so adversaries can easily use them to identify records. Clearly there is a need to select appropriate features for anonymization to optimize the balance between privacy protection and utility of data. The selection method considers for each feature both its relationship with the target variable as well as privacy risks and suggests a subset of features to be anonymized. This process is explained in detail in section 3.3.

Algorithm 1 next divides the data set into subsets based on the class label. This helps improve the utility of anonymized data because one does not want to blur the boundary between records with different class labels. If D does not have a class label then the whole data set will be treated as a single subset (class).

For each subset D_i, Algorithm 1 divides it into clusters such that each cluster has at least k records, such that the records in the same cluster have similar values. In order to preserve differential privacy, it is important to make sure that adding or deleting a record to D_i has a limited impact on resulting clusters. Our previous work using a condensation method for anonymization in [3] does not satisfy this property so it may not lead to differential private output. Instead, in Algorithm 1 we use an insensitive microaggregation method [24] which ensures that adding a record can only cause limited changes to resulting clusters. Details of our insensitive microaggregation process are described in Section 3.2.

Once these clusters are generated, Algorithm 1 computes the mean of numerical features that need to be anonymized in each cluster, and adds Laplace noise to the mean values to ensure differential privacy. Each record in a cluster is then replaced with another one containing these perturbed mean values on those features. For categorical features, the algorithm uses exponential mechanism to return the most frequent value within a cluster [20]. Note that there is a small probability that a value other than the most frequent value will be returned to ensure differential privacy.

One issue of Algorithm 1 is that the input parameters ϵ and k have significant impact on the output so they need to be tuned appropriately to optimize the balance between privacy protection and data utility. This is an important problem to solve, and we propose an automatic parameter tuning method to address this issue.

The rest of this section provides details of each step in Algorithm 1. Section 3.1 describes anonymization of IP addresses. Section 3.2 presents the insensitive microaggregation method and proves differential privacy. Section 3.3 describes our proposed method that selects the features to be anonymized based on properties of the data set D. Section 3.4 describes the automatic parameter tuning method.

3.1 Anonymization of IP/MAC addresses

Our method first automatically detects IP/MAC addresses and applies prefix-preserving anonymization to the address fields. The digits of each address are split into the leading digits and the remaining digits. The split is decided by the type of the address. The leading digits (network part of the IP address) are anonymized by random permutation. The remaining digits (host part of the IP address) are treated as a regular field and are anonymized along with other fields. If the prefix length of an IP address is already specified (e.g., in the CIDR notation the address ends with /x where x is length of prefix), then our method uses the specified length. If the prefix length is not specified, we use the first 24 bits as prefix for IPv4 addresses as that is the most commonly used prefix length [3]. For an IPv6 address field, our method first decides whether it is unicast or multicast by checking whether the first 8 bits are all ones (signifying multicast) or not (unicast). For unicast, usually the first 48 bits need to be preserved because they represent routing prefix and subnet ID, and the last 80 bits should be anonymized because they identify a specific computer [1]. For multicast, usually the first 96 bits are the prefix.

After anonymizing the network part, the addresses are then clustered into k clusters based on the host part of the address. The average value of every 8 bits in the host part of IP addresses (within the same cluster) is calculated. Then the host address digits in each cluster are replaced using this computed average value.

3.2 Insensitive Microaggregation

Algorithm 2 presents the insensitive microaggregation (IMA) approach, which ensures that the addition of a record has a minimal effect on clusters. The algorithm first computes a boundary point P, where P takes the minimal value of each feature to be anonymized. All records are sorted by their distance to P. The method then forms a cluster with the K points closest to P. These points are removed from the data set. This process repeats until no more than $2K$ points are left. Finally, the remaining points form the last cluster.

Figure 1 illustrates the basic idea behind this method on an example dataset containing nine points (records). The circles identify clusters, and $K = 3$. The triangle represents the boundary P with minimal value on each dimension. The nine points are sorted by their distance to P and a first cluster is generated containing the three points closest to P. Then, these points are removed from the data set and a second (and finally a third) cluster is generated with the next three closest points.

Algorithm 2 Insensitive Microaggregation

procedure IMA(D_i,ϵ,k)
  Compute a boundary node P which takes minimal value in D_i on every feature to be anonymized
  for Each record $x \in D_i$ **do**
    Compute distance between x and P
  end for
  Sort records in D_i by their distances to P
  while D_i contains at least $2k$ records **do**
    Form a new cluster C_{ij} by taking the first k records in D_i with shortest distance to P
    Delete these records from D_i
  end while
  if D_i is not empty **then**
    Form a new cluster with the remaining records in D_i
  end if
end procedure

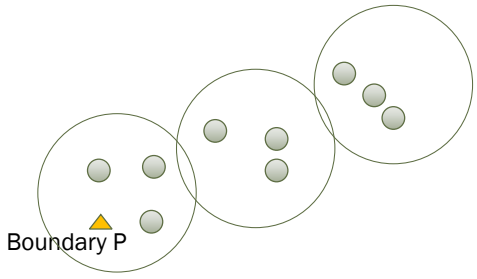

Fig. 1. Example of insensitive microaggregation

In order to prove that our algorithm is differentially private, it is sufficient to show that the addition of a record z has limited impact on the result. Suppose a point z is added, and its distance to P is in between x_i and x_{i+1}. Suppose these points are sorted by their distance to P as $x_1, x_2, \ldots, x_n$. Clearly the first cluster around P is formed by $x_1, \ldots, x_k$, and the next cluster is formed by $x_{k+1}, \ldots, x_{2k}$ and so on. Only clusters formed after x_i are affected by the addition of z and the difference between such a cluster before and after the addition of z will be at most one point. This allows us to bound the sensitivity (i.e., the maximal impact) of adding z on cluster mean by $\frac{A_{max}}{k}$ where A_{max} is the maximal value of a feature and k is the minimum cluster size. There are at most n/k clusters affected. As a result, a Laplace noise with mean zero and standard deviation $\frac{A_{max}n}{k^2\epsilon}$ can be added to ensure ϵ differential privacy.

3.3 Selection of Features for Anonymization

The problem of selecting a subset of features for anonymization is similar to the problem of feature selection. However, the former needs to consider both privacy risks and the subsequent analysis tasks while the latter only considers accuracy of the analysis task.

In general, if we know the benefits (better privacy protection) and costs (loss of accuracy in subsequent analysis) of anonymizing each feature, the problem of recommending the best features for anonymization is still NP-hard because it is equivalent to the Knapsack problem. Therefore, we use a greedy method which ranks each feature based on its privacy risks and usefulness for subsequent analysis. We present three methods to recommend features for anonymization, described next.

CUS **Method:** This method measures how unique a value of a record for a particular feature is, within the context of the entire set of values of that feature. The rationale behind this method lies with the scarcity of each value in the feature: the fewer times a value is repeated in the feature, the easier it is to re-identify a specific record. Formally, for each feature x_i, a Context uniqueness Score (CuS) is calculated for each distinct value v_{ij} of x_i where $CuS(v_{ij})$ equals the percentage of records with v_{ij}. The context uniqueness score for the feature x_i is defined in Equation 3.2:

$$CuS(x_i) = \min_{v_{ij} \in x_i} CuS(v_{ij}) \tag{3.2}$$

$CuS(x_i)$ measures the re-identification risk of feature x_i. The lower the CuS value, the higher the risk or re-identification. For example, a feature with rare values will have a lower CuS score than a feature without rare values because the rare value has lower frequency in the data set and higher re-identification risks because adversaries can use the rare value to infer the identity of a record.

Mutual Information Method: This method utilizes mutual information, which is a measure of dependence between two variables, in essence, it identifies how much information we can obtain about one variable by observing the other variable. In order to measure utility, we use mutual information between the feature and the target variable that needs to be predicted in the subsequent analysis because it works for both discrete target variables (e.g., attack or not attack) and numerical target variables (e.g., the performance of transferring large files over WLAN). Let x_i be a feature and y be the target variable. Mutual information $I(x_i, y) = H(x_i) - H(x_i|y)$ where $H(x_i)$ is the amount of entropy of x_i and $H(x_i|y)$ is the conditional entropy of x_i given y. The Mutual Information method simply ranks features by their mutual information in ascending order. Features with the lowest mutual information will be recommended for anonymization because they contribute less to predicting the target variable.

CUP **Method:** Finally, we also propose a method that builds on the previous two mentioned methods. It combines the Context Uniqueness Score and the Mutual Information score. We call it a Combined Utility-Privacy (CUP) score for X and it is defined in Equation 3.3:

$$CUP(x_i) = I(x_i, y) \times CuS(x_i) \tag{3.3}$$

This method uses CUP to sort features in ascending order. The top ranked features have a high re-identification risk, since they have either a low CuS score or low mutual information with the target variable. Therefore, these features are recommended to be anonymized. Regardless of the proposed recommendations by our method, the data owner always reserves the right to review these recommended features before making a final decision of which features to anonymize.

3.4 Automatic Parameter Tuning

Our anonymization method requires two important parameters: the privacy budget ϵ used in differential privacy and the minimal cluster size k. When we increase ϵ, the noise added in Algorithm 2 becomes less, so the degree of privacy protection becomes lower and the analysis results over anonymized data are more accurate. When we decrease ϵ the opposite happens.

The impact of k is more subtle. When we decrease k, our method generates more clusters, which should lead to less privacy protection and more accurate results. However, since the noise added in Algorithm 2 is reciprocal to k, more noise will be added, increasing privacy protection and generating less accurate results. Therefore, the overall impact of k is less certain.

The problem of automatically tuning and identifying parameter values is not new. There has been a lot of work on tuning hyperparameters [29]. The most popular methods include grid search, random search, and Bayesian optimization. Grid search picks a few values for each parameter and then exhaustively tries all combinations of these values, so it is quite expensive if there are many parameters or many possible values. Random search each time checks a few random combinations and selects the best one. It is more applicable when there are many parameters. Bayesian optimization is a popular optimization strategy for a black-box objective function that does not assume any functional forms. It is usually employed to optimize expensive-to-evaluate functions [13]. In parameter tuning Bayesian optimization is used to find the parameter setting that optimizes an objective function (e.g., the accuracy).

Overall, none of the above methods if used by themselves is sufficient to address parameter tuning for anonymization. Specifically, in anonymization of data sets one has two objectives to optimize: the accuracy of subsequent analysis conducted on anonymized data and the degree of privacy protection. Both objectives can be measured using different metrics. For example, we can use true positive rate (TPR) and false positive rate (FPR) for data used in cyber security. However, it is not trivial to estimate TPR and FPR because they are data dependent. So getting an accurate estimation of these metrics often requires several experimental attempts. To describe our tuning approach we will use a cyber security dataset as an example and will calculate values for k and ϵ parameters. An important part of this process is to provide a metric that measures the effectiveness of combinations of k and ϵ values. Next, we describe a way to measure the quality of a specific combination of k and ϵ parameter values and then we propose a hybrid method that selects the k and ϵ parameter values.

Measuring quality as a combination metrics: Within the scope of a cyber security dataset, we use TPR and FPR for attack class, weighted F1 score for both attack and normal classes, and privacy level to measure the quality of a combination. TPR and FPR are more commonly used than accuracy in cyber security because cyber security data sets are often very imbalanced and accuracy is not a good measure for imbalanced data. TPF and FPR only measure performance of the attack class. F1 score measures performance on both classes and combines both precision and recall. Since ϵ only measures worst-case privacy, we use a privacy level metric in [2] because it is easy to interpret, and represents on average how close the original value lies to the anonymized value. Suppose the original value of a feature x_i can be estimated in the range of $[v_1, v_2]$ with 95% confidence, this metric computes a privacy level at $(v_2 - v_1)/std(x_i)$ where $std(x_i)$ is the standard deviation of the original feature. It then averages this privacy level over all anonymized features. For example, a privacy level of two means the original value can be estimated at an interval twice the size of the standard deviation. In essence, the longer the length of the confidence interval, the higher degree of privacy protection. We also take a log of the privacy level because unlike TPR, FPR, and F1 score, the privacy level is not bounded within $[0, 1]$ and a privacy level greater than two is often sufficient because the original data can be only

estimated to the range plus or minus one standard deviation from anonymized data. Hence, by consolidating all these metrics together, we construct Equation 3.4 to measure the overall quality Q.

$$Q = w_1 \times TPR - w_2 \times FPR + w_3 \times \text{F1 score} + w_4 \times \log(\text{privacy level}) \tag{3.4}$$

In practice users can set values for weights w_1, w_2, w_3, w_4 based on their preferences. We set w_1, w_2, w_3 to 1 and w_4 to 0.1 for experiments in Section 4 because our experiments used intrusion detection data where higher priorities should be given to accuracy of intrusion detection using anonymized data than to privacy protection.

Hybrid Algorithm: Based on these observations, we propose a hybrid approach described in Algorithm 3. It is called hybrid because it uses both the crude grid search and the greedy search. The basic idea is to first perform a crude grid search to identify a few promising combinations, and then uses greedy search around the area in each promising combination. This allows us to find near optimal combination in very few number of attempts.

The algorithm takes input a data set D, a set of selected features F to anonymize, an initial grid size s (we set s between three and four in this paper), two range parameters δ_k and δ_ϵ which defines the area to search around a promising combination, and two stopping thresholds: t_q for minimal improvement and t_δ for minimal range.

Algorithm 3 Hybrid method for parameter tuning

procedure HybridTuning(data set D, feature set F to anonymize, initial grid size s, range δ_k and δ_ϵ quality threshold t_q and range threshold t_δ)
  Choose minimal and maximal value for ϵ and k
  Generate a grid with $s \times s$ cells over ϵ and k
  for Each combination of ϵ_i and k_j in the initial grid **do**
    D'=Anonymize_With_IMA (D,ϵ_i,k_j)
    Train a model using D' and compute score(ϵ_i, k_j) using Equation 3.4
  end for
  Select a few combinations with best score
  for Each selected combination of ϵ_u and k_v **do**
    repeat
      if not the first iteration $k_v = k'_v$ /*replace start point with current local optimal*/
      Fix ϵ at ϵ_u, do a size s grid search of k in the range of $[\max\{k_v(1-\delta_k), k_{min}\}, \min\{k_v(1+\delta_k), k_{max}\}]$ and find a local optimal k'_v (i.e., with highest score)
      $\delta_k = \delta_k/2$
    until score(k'_v, ϵ_u)-score(k_v, ϵ_u) $< t_q$ or $\delta_k < t_\delta$ /*stop if improvement is less than a threshold or the range reaches minimal threshold*/
    repeat
      if not the first iteration $\epsilon_u = \epsilon'_u$ /* replace start point with local optimal */
      Fix $k = k'_v$, do a grid search to find a local optimal ϵ'_u in the range of $[\max\{\epsilon_u/(1+\delta_\epsilon), \epsilon_{min}\}), \min\{\epsilon_u(1+\delta_\epsilon), \epsilon_{max}\}]$
      $\delta_\epsilon = \delta_\epsilon/2$
    until score(k'_v, ϵ'_u) -score(k'_v, ϵ_u) $< t_q$ or $\delta_\epsilon < t_\delta$
  end for
  Return the pair of (ϵ', k') with the highest score
end procedure

Our algorithm first decides the minimal and maximal value of each parameter as follows. Let these values be $\epsilon_{min}, \epsilon_{max}, k_{min}$, and k_{max}. For k we select minimal value of 20 for data sets with fewer than one thousand records and a value of 100 otherwise. Smaller k will cause the noise added in Algorithm 2 to be too large. We set the maximal

value of k at n/c such that at least c clusters will be generated (so cluster size will be at most n/c) to make sure the anonymized data captures enough properties of the original data. We use $c = 200$ in this paper. Note that records that fall within the same cluster will have the same value on each anonymized feature.

For ϵ we let users select its minimal and maximal value. In this paper we used a minimal value of 0.01 and a maximal value of 1.

The algorithm then generates a $s \times s$ grid in the allowed ϵ and k ranges. The best few combinations are selected; we choose one combination in this paper. For k the algorithm divides it in equal-sized grids. For ϵ the algorithm divides it in equal-sized grids in logarithmic scale because ϵ can take very small values.

For each selected combination (ϵ_u, k_v), the algorithm first fixes the value of ϵ, and does a local search on k. Note that this step is greedy because the algorithm ignores other ϵ values. We also optimize k first because we found that it has a larger impact on results compared to ϵ.

The algorithm uses a step s grid search in the local search step on an interval of $[\max\{k_v(1-\delta_k), k_{min}\}, \min\{k_v(1+\delta_k), k_{max}\}]$ where the k value falls between $k_v(1-\delta_k)$ and $k_v(1+\delta_k)$ and is bounded by the previously selected minimal and maximal value of k. For example, if $k_v = 500$ and $\delta_k = 0.2$, it will search in the range of $[400, 600]$. Grid search is used here because the relationships between k and quality metrics are more complex and not necessarily monotonous. But this grid search is only conducted on only one parameter so it is still quite efficient.

The algorithm then replaces k_v with local optimal k'_v and δ_k with $\delta_k/2$. The range δ_k is halved to focus on smaller areas. This process repeats the local grid search on k until the improvement is lower than a threshold t_q or the range δ_k is less than t_δ.

The algorithm then fixes k at the found local optimal k'_v and uses a grid search to find a local optimal value for ϵ in the neighborhood of $[\max\{\epsilon_u/(1+\delta_\epsilon), \epsilon_{min}\}), \min\{\epsilon_u(1+\delta_\epsilon), \epsilon_{max}\}]$. This process is similar to the step in finding a local optimal k. But the algorithm searches in the range of $\epsilon/(1+\delta_\epsilon)$ to $\epsilon(1+\delta_\epsilon)$ because ϵ is divided into grids in logarithmic scale. For example, if $\epsilon_u = 0.1$, $\delta_\epsilon = 1$, the search takes place in the range of $[0.05, 0.2]$.

Again a one-dimensional grid search is used, because although each individual quality metric is monotonous with respect to ϵ, the overall score in Equation 3.4 may not be monotonous. Specifically, when ϵ increases, the accuracy metrics (TPR, FPR, F1 score) should yield better results. However, the privacy level should decrease because less noise is added to the original data. As a result, it is unclear whether the combined score in Equation 3.4 will increase or decrease.

Finally the algorithm returns the best combination found in all searched combinations.

Complexity Analysis: Algorithm 3 searches a total of $O(s^2 + s \log \max\{\delta_k/t_\delta, \delta_\epsilon/t_\delta\})$ combinations because the initial grid search checks s^2 combinations and then the local search only checks $O(s)$ combinations in each iteration, and can have at most $\log \max\{\delta_k/t_\delta, \delta_\epsilon/t_\delta\}$ iterations because each time the two range parameters are halved. The value of s can be set quite low compared to a typical grid search so in practice our hybrid algorithm is more efficient. Quality wise we observed through experiments that Algorithm 3 produces results that are comparable to a grid search with much larger number of combinations.

4 EXPERIMENTAL RESULTS

In this section we discuss the experiments that we conducted in order to evaluate our anonymization approach and compare it with other existing tools or methods. The goal of anonymization is not only to obscure a data set, but also to ensure that an analysis of the anonymized data set is as close as possible to the one performed on the original non-anonymized data set. Therefore we also compare our results with those on the original data set. Furthermore, we evaluated the capability of our approach to thwart injection attacks on the data set; we also describe our evaluations on

selecting and recommending the most appropriate features in a data set for anonymization. Finally, we evaluated the parameter setting and tuning of our main Anonymization_With_IMA approach. In order to conduct all these evaluations, we needed to use specific data sets. Since network data is used for many purposes and in several domains, such as in cyber security, it makes sense to use cyber security data sets to conduct experiments and evaluate our proposed methods within the scope of this research.

4.1 Setup

Data sets: The data sets that we used in our evaluation are: PREDICT [15] and UNSW-NB15 [23]. We chose these datasets because they contain network trace information for cyber security purposes. In addition, they are labeled and contain a variety of cyber attacks, thus enabling a more accurate subsequent analysis. The PREDICT dataset contains 500K rows and 13 fields such as IP information and other flow statistics. UNSW-NB15 has 42 numerical fields and we used about 120K rows. Detailed attributes in UNSW-NB15 dataset allow us the opportunity to experiment with the feature selection process, such as TCP connection setup time. Both data sets are split into around 70% training and 30% testing.

Methods: We compared the proposed Anonymization with Insensitive Micro Aggregation (Anonymize_with_IMA) method against two categories of existing anonymization methods. The first category includes general purpose data anonymization tools, while the second category is more specific to network data anonymization tools. Regarding general anonymization tools, we selected ARX [12] because it is a popular anonymization tool and implements multiple anonymization methods and privacy models. We selected two main anonymization methods implemented in ARX: 1) generalization and suppression, and 2) micro aggregation. Both methods enforce k-anonymity. Note that our method enforces both k-anonymity and differential privacy. For the case of generalization and suppression, ARX allows users to create a generalization hierarchy for each quasi-identifier attribute. For example, packet sizes can be generalized into intervals and higher level intervals can be divided into smaller intervals at a lower level. ARX then searches for the optimal combination of generalization of all quasi-identifier attributes to satisfy k-anonymity and at the same time minimizes loss of utility. The micro aggregation method in ARX is similar to the clustering step of our method that divides data into clusters and then replaces the quasi-identifier attributes of records in each cluster with their mean. But there are two major differences: 1) the method in ARX may mix records with different class labels (e.g., malicious and normal traffic) in the same cluster, which leads to lower utility; 2) ARX does not enforce differential privacy whereas our method does.

The second category of tools refers to those that are specific for network data. Examples of such tools include PCAPLIB [18], Crypto-PAn [28], TCPANON [14], SCRUB-TCPDUMP [17], and TRACEWRANGLER [4]. All of them use prefix-preserving anonymization methods on IP addresses. For the other fields, they use a number of simple masking/obscuring techniques including:

(1) Random permutation (i.e., values of a field are randomly permuted within the data set);
(2) Black marker (a field value is replaced with a constant);
(3) Truncation (the last few digits of a field are removed).

In order to enable a comparative analysis between the above tools and ours, we implemented these three techniques, applying prefix-preserving anonymization for IP addresses and then using one of these three techniques for the other fields. Since there is a degree of randomness in Anonymize_With_IMA Algorirhtm (because it adds randomly generated

noise to cluster means), we experimented by executing Anonymize_With_IMA five times and reported the average of the results. We did not notice much variation of results between the different executions.

Metrics: In our experiments we measure both the degree of privacy protection and the preservation of data properties (utility) for subsequent analysis. As other methods do not enforce differential privacy, we use the confidence interval metric [2] as a privacy metric, which is the length of the 95% confidence interval of the difference between the original data and the anonymized data (the larger the interval the higher the privacy protection). Since different fields have different ranges, we normalized the confidence interval by the standard deviation of each column. For data utility we use the weighted F1 score (for all classes). Cyber security data is often unbalanced (a data set contains way more normal records than attack records) therefore, the F1 score is a better measure than accuracy. In addition, we use true positive rate (TPR, i.e., the fraction of attacks being detected) and false positive rate (FPR, i.e., the fraction of false alerts) for the attack class. TPR and FPR are important metrics in cyber security. For completeness, we also include the True Negative Rate (TNR, fraction of normal instances being predicted as normal) and False Negative Rate (FNR, fraction of attacks being missed). TNR = 1-FPR, and FNR = 1-TPR. In general, better results are those that yield higher values of F1 score, TPR, TNR, and lower values of FPR and FNR.

4.2 Comparative Analysis with Other Anonymization Methods

Both the UNSW-NB15 and the PREDICT cyber security data sets were used to conduct the same experiment. First, each data set was anonymized. Then, an intrusion detection analysis was performed on it to identify cyber attacks using a training set (approximately 70%) and a testing set (approximately 30%) of the data set. Subsequently, we conducted the same intrusion detection analysis on the original non-anonymized data set, and compared the anonymized analysis results to the non-anonymized ones.

Since injection attacks can be applied to many fields, such as packet size, or timestamp, we decided that all methods should anonymize all fields except the class label (attack or not attack). This approach enables a straight-forward comparison of the tools.

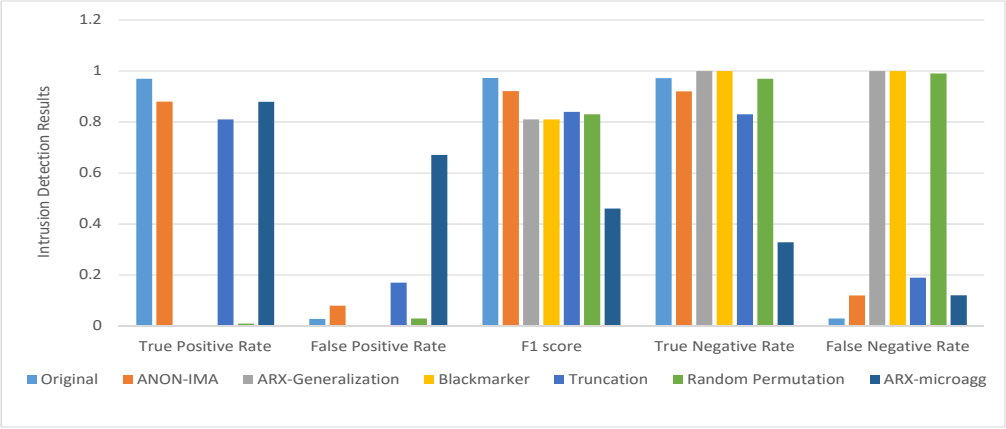

Fig. 2. Intrusion detection results using several anonymization tools/methods on the UNSW-NB15 data set using K-NN classifier. **Tools/Methods** (from left to right): Original data, Anon-IMA (Anonymize_with_IMA), ARX-Generalization, Blackmarker, Truncation, Random Permutation, ARX Microaggregation. **Metrics:** True positive rate, False positive rate, F1 score, True negative rate, and False negative rate

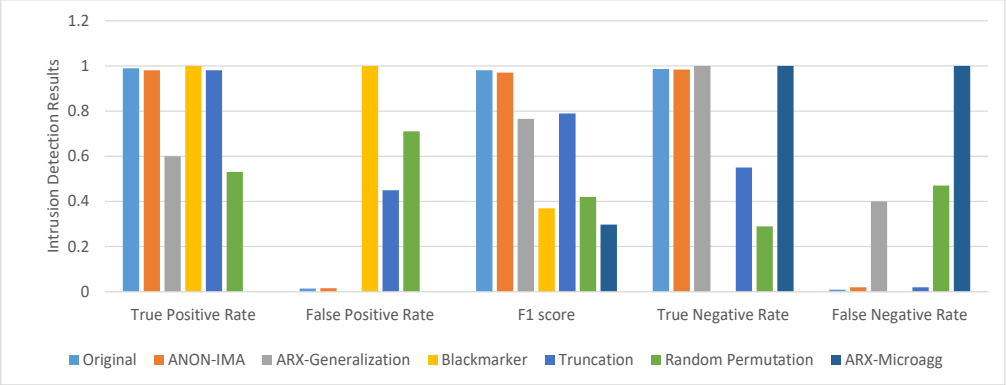

Fig. 3. Intrusion detection results using several anonymization tools/methods on the PREDICT data set using K-NN classifier. **Tools/Methods:** Original data, Anon-IMA (Anonymize_with_IMA), ARX-Generalization, Blackmarker, Truncation, Random Permutation, ARX Microaggregation. **Metrics:** True positive rate, False positive rate, F1 score, True negative rate, and False negative rate

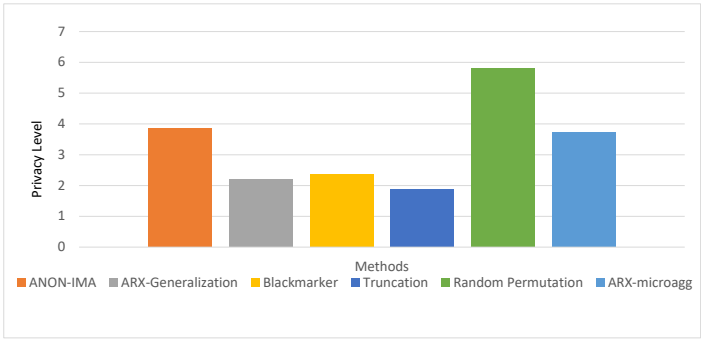

Fig. 4. Privacy levels for various methods on UNSW-NB15 data

We set the privacy budget to 0.1 and cluster size $k = 500$ in our solution. We used the default setting for all other methods, and set $k = 500$ in ARX.

Figure 3 and Figure 2 report the values of F1 score, TPR, FPR, TNR, and FNR of intrusion detection experiments on data anonymized by various methods for the PREDICT and the UNSW-NB15 datasets respectively. K-NN is used to predict whether a test instance is an attack because it gives the highest accuracy on these datasets. First, we compare the analysis results using our anonymization approach with those on the original data set. The results show that our approach better preserves data properties important for subsequent analysis. For the PREDICT data, TPR using our approach reaches 98% and weighted F1 score is 97%, both within 1% of the results on original data. For UNSW-NB15, the TPR of our method is around 88% and overall F1 score is 92%, slightly lower than those on the original data, but still higher than the other anonymization methods. The FRP of our approach is also around 1.6% for the PREDICT data and 8% for the UNSW-NB15 data, slightly higher than the FPR on the original data. The TNR of our approach is around 98% for PREDICT and 92% for UNSW-NB15. The FNR of our aporoach is around 2% for PREDICT and 12% for

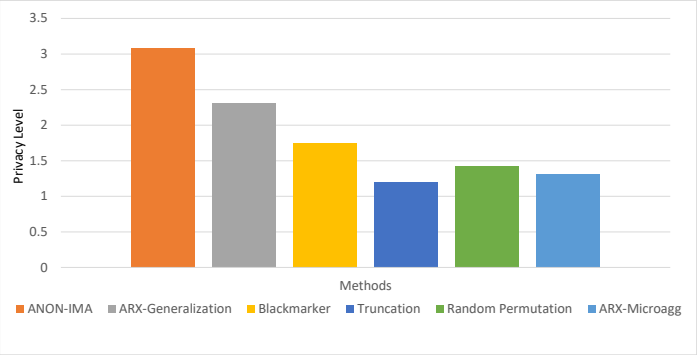

Fig. 5. Privacy levels for various methods on Predict data

UNSW-NB15. Overall based on these experiments the Anonymization_With_IMA method achieves performance similar to the original data, and at the same time it preserves data utility.

Next we compare our Anonymization approach with the other tools/methods using various metrics.

In terms of true positive rate and false negative rate, our approach along with Truncation and ARX with micro aggregation produced acceptable results on the UNSW-NB15 data set. All other methods gave near zero TPR and close to 100% FNR. On the PREDICT data set, our approach, Blackmarker and Truncation showed good results (near 100% TPR and near zero FNR). Truncation seems to work well for both data sets because it still preserves enough useful information by just removing the least significant digits of attributes in the data set. ARX with micro aggregation works well for UNSW-NB15 but does not work well for PREDICT, possibly because the PREDICT data set is more skewed with very few malicious instances. Since ARX with micro aggregation may put together malicious instances with normal instances in the same cluster, it may introduce a larger distortion for skewed data sets.

In terms of false positive rate and true negative rate, all methods work relatively well on the UNSW-NB15 data set except ARX with micro aggregation, which produces around 67% FPR and less than 33% TPR. This happens probably due to the micro aggregation method in ARX putting together malicious instances with normal instances. For the PREDICT data set, our approach and the two ARX methods work relatively well, however, Truncation, Blackmarker and Random permutation lead to a high FPR and low TNR.

In terms of F1 score, our Anonymization_With_IMA approach achieves F1 score quite close to the original data on both data sets. On the PREDICT data set, Truncation, and ARX with generalization achieve an F1 score near 80%, lower than that of our approach. Other methods have an F1 score around 40% or less. On the UNSW-NB15 data, the remaining methods except ARX with micro aggregation produce an F1 score around 80%, which is acceptable but lower than that of our Anonymization_With_IMA.

Overall, Anonymization_With_IMA achieves good performance across all metrics on both data sets. In addition, the analyses difference conducted using the anonymized and the original data is also small. The remaining methods exhibit a lower performance compared to Anonymization_With_IMA on some metrics.

Next we briefly explain the variations of the results accross the different tools/methods. Both datasets (also true for many network datasets) contain many fields that need to be anonymized due to injection attacks. Since ARX with generalization/suppression generalizes each field independently, it often has to generate extremely low resolution data

to provide enough privacy protection. For example, suppose each anonymized field is generalized into two distinct values. The number of possible combinations of 40 fields will be over one trillion but the number of records in the dataset is far smaller. So many anonymized records will have unique values which can be easily re-identified. As a result ARX generalizes the majority of fields to just one distinct value. This leads to poor accuracy as K-NN using ARX anonymized data predicts all UNSW-NB15 test data as normal, missing all the attacks (it has zero TPR and FPR, and F1 score is high simply because 80% of records are normal). For PREDICT its TPR is also only 60%, missing many attacks.

ARX with micro aggregation puts together malicious instances with normal instances in the same cluster. This leads to high false positive rate and low F1 score for UNSW-N15B. Its performance on PREDICT is similar, leading to K-NN predicting all testing instances as normal, thus missing all malicious instances. Anonymization_With_IMA avoids this problem by running clustering for each class separately. It also enforces differential privacy.

Blackmarker basically reduces each anonymized field to a constant. So K-NN just predicts every record to be an attack (the case of PREDICT where TPR=FPR=1) or normal (the case of UNSW-NB15 where TPR=FPR=0).

Random permutation does not preserve data properties since the correlation between features and the target variable is lost. In the UNSW-NB15 data it classifies almost all test data as normal. This leads to near zero TPR meaning it missed all the attacks. The F1 score is still 0.83 simply because most test instances are normal. The PREDICT data has more balanced test data. Random permutation incorrectly classifies many attacks and normal instances and has low TPR, high FPR, and low F1 score.

Truncation gives more reasonable results because it preserves some portion of original values (only the last few digits are lost). But it still distorts fields with small values (e.g., if it removes the last 3 digits then any number from 0 to 999 will become zero). This behavior explains the results (e.g., in PREDICT data its FPR is quite high and in UNSW-NB15 its TPR is only around 81%). In addition as shown in Figure 5 and Figure 4, its privacy level is significantly lower than the privacy level of our method. It is also vulnerable to injection attacks as it provides no strong privacy protection as differential privacy.

Figures 4 and Figure 5 show privacy level (in terms of confidence interval). Our approach achieves a similar or higher level of privacy than other methods except random permutation on UNSW-NB15, but random permutation does not preserve utility (it missed all attacks in the UNSW-NB15 dataset).

Table 1 and Table 2 show a sample of a few records in the UNSW-NB15 original and anonymized data. It can be seen that IP addresses with the same prefixes in the original data still have the same prefix after anonymization; hence, the anonymization process preserves prefixes of IP addresses. The values of numerical features are perturbed after anonymization but they are still in a similar order. For example, the first and the fourth record have larger src bytes than the other two records in the original data set and the same is true in the anonymized data set. This helps in conducting analysis using the anonymized data because properties of the original data are preserved to a large extent.

Results of Simulated Injection Attacks: We conducted experiments to examine whether the proposed Anonymization_With_IMA algorithm can thwart injection attacks. Injection attacks insert packets or flows of packets with certain patterns (e.g., specific destination port numbers, with specific delay between packets) which are incorporated into the original data. When attackers obtain access to the anonymized data, they can identify these injected patterns, and then may uncover original data based on relationships between injected and original records. For UNSW-NB15, we randomly created a total of 750 injection flows, divided into sets of 150 flows, each set for one of the five different patterns proposed in [6]. Then we try to identify the injected patterns from the anonymized data using K-NN-search. The K-NN-search algorithm finds the nearest neighbor of the injected pattern from the anonymized dataset. It results in the row number of the nearest neighbor. If the row number is a match with the original row number of the injected

Table 1. Original Data

Src IP	Dest IP	Duration	Src Bytes	Dest Bytes
59.166.0.0	149.171.126.9	0.036	528	304
59.166.0.6	149.171.126.7	0.0012	146	178
59.166.0.5	149.171.126.5	0.0012	132	164
175.45.176.2	149.171.126.16	0.17	8168	268

Table 2. Anonymized Data

Src IP	Dest IP	Duration	Src Bytes	Dest Bytes
135.126.163.4	71.29.51.5	0.034	5257	300
135.126.163.8	71.29.51.7	0.0019	263	211
135.126.163.2	71.29.51.1	0.0018	298	189
147.171.114.1	71.29.51.15	0.58	26070	257

data, then the attack is successful. We compared the performance of our method with the original dataset, general truncation and ARX with generalization, and measure the recovery rate, which is the percentage of recovered instances. Lower recovery rate means that the anonymization method is more robust and can defend against injection attacks. For both our Anonymization_With_IMA Algorithm and the ARX with generalization, we used $k = 50$ for k-anonymization method. Table 3 shows the results of the injection attack re-identification.

Table 3. Recovery rate of injection attacks for various anonymization methods with 95% confidence interval

Original data	Truncation	ARX-Generalization	Anonymization_With_IMA
85.33% ±0.57%	3.60% ±0.029%	14.40±0.095%%	0.27% ±0.0095%

Table 3 shows the recovery rate of injection attacks with 95% confidence interval over various anonymization methods. We observe that our method works better than the other methods in thwarting injection attacks.

4.3 Evaluation of Feature Selection Method

In addition, we conducted experiments to identify whether our selection methods can assist in identifying which features are good candidates to anonymize in a given dataset. Therefore, we compared three selection methods to select features to anonymize as well as a baseline that anonymizes all features.

(1) *CUP*: this method uses the combined utility-privacy score in Equation 3.3 to rank features. Only the top ranked features are anonymized.
(2) *CUS*: this method uses only the privacy score in Equation 3.2 to produce its output.
(3) Mutual Information: this method only uses mutual information term in Equation 3.3.

All three selection methods selected 20 out of 42 features for anonymization. We set $\epsilon = 1$ and $k = 500$ in this set of experiments. Table 4 shows the results on UNSW-100K data set. The weighed score is computed using Equation 3.4. We do not report results on PREDICT because on that data set even anonymizing all columns produces high values in TPR, F1 score and a very low FPR, making feature selection less important.

The results show that all three selection methods have higher TPR, and F1 score than anonymizing all columns. So anonymizing fewer columns does improve utility of the anonymized data. Anonymizing all columns does have slightly higher privacy level, but that is offset by reduced utility as shown in its lower weighted score compared to the three selection methods.

Table 4. Experiment results for selecting candidate features for anonymization on the UNSW-100K data set

Method	TPR	FPR	F1	Privacy Level	Weighted Score
All Columns	0.888	0.091	0.914	3.7555	1.768
CUP	0.958	0.05	0.95	3.17	1.908
CUS	0.959	0.05	0.955	3.22	1.914
Mutual Information	0.958	0.05	0.951	3.29	1.91

For this data set all three methods perform quite well. The performance of *CUP* and Mutual Information are similar because many features have unique values and thus their *CUS* scores are similar. Since *CUP* score equals *CUS* score multiplied by Mutual Information, *CUP* and Mutual Information method share many top ranked features. We also observe that *CUS* exhibits very good results. There are two possible reasons for this: First, around 60% of features are selected by both *CUS* and *CUP*, because many features having high Mutual Information also have rare values. Second, our anonymization method is quite effective at preserving data properties important for subsequent analysis so even after anonymizing the features identified by *CUS* subsequent analysis is still quite accurate.

Table 5 shows the top 10 features ranked by *CUP*. It can be seen that many of these features can help identify attacks. For example, synack is considered an important feature [22] because a longer TPC connection setup time often indicates denial of service attacks. Most of these features also have integer or float values, meaning they are likely to have more unique values with high privacy risks.

Table 5. Top 10 features ranked by *CUP*

Feature Name	Data Type	Description
synack	Float	TCP connection setup time, between the SYN and the SYN_ACK packets.
is_sm_ips_ports	Binary	If source and destination IP addresses equal and port numbers equal then this variable takes value 1 else 0
smeansz	integer	Mean of the flow packet size transmitted by the src
ct_state_ttl	integer	No. for each state according to specific range of values for source/destination time to live
ct_flw_http_mthd	Integer	No. of flows that has methods such as Get and Post in http service
dmeansz	integer	Mean of the fow packet size transmitted by the dst
ct_srv_dst	integer	No. of connections that contain the same service and destination address in 100 connections according to the last time
dwin	integer	Destination TCP window advertisement value
Spkts	integer	Source to destination packet count
Dpkts	integer	Destination to source packet count

4.4 Evaluation of Hybrid Parameter Tuning Method

We compared our hybrid parameter tuning method (Algorithm 3) with a standard grid search approach. The hybrid method starts with a 3×3 grid search and then performs a greedy search around the best combination found in the initial grid search. We set range $\delta_k = 0.2$, $\delta_\epsilon = 1$, performance improvement threshold $t_q = 0.001$, range threshold $t_\delta = 0.05$. The standard grid search algorithm has a larger grid sized 6×6. In this experiment we report results when all features are anonymized. When conducting experiments after using the feature selection methods we also observe similar results.

Table 6 shows the results of each stage of the Hybrid algorithm on UNSW-100K data. The results on PREDICT are similar. The weighted score is computed based on Equation 3.4. The initial grid search checked 9 combinations and the best combination found is $k = 500, \epsilon = 0.1$. The greedy search on k checked 3 combinations and found no improvement. The greedy search on ϵ found some better combinations so it stopped after two iterations of grid search (i.e., six combinations). The final combination returned is $k = 500, \epsilon = 0.17$.

Table 6. Hybrid Parameter Tuning Results on UNSW-100K

Stage	# of Combinations	Best (k, ϵ)	TPR	FPR	F1	Privacy Level	Weighted Score
Initial Grid Search	9	$k = 500, \epsilon = 0.1$	0.877	0.079	0.921	3.848	1.777
Greedy search on k	3	$k = 500, \epsilon = 0.1$	0.877	0.079	0.921	3.848	1.777
Greedy search on ϵ	6	k=500, $\epsilon = 0.17$	0.914	0.098	0.912	3.785	1.786

Table 7. Hybrid Parameter Tuning vs. Grid Search on UNSW-100K

Algorithm	# of Combinations	Best (k, ϵ)	TPR	FPR	F1	Privacy Level	Weighted Score
Hybrid	18	k=500, $\epsilon = 0.17$	0.914	0.098	0.912	3.785	1.786
Grid	36	$k = 500, \epsilon = 0.1$	0.877	0.079	0.921	3.848	1.777

Table 7 shows the best combinations observed using the hybrid method compared to the standard grid search method. The results show that the Hybrid approach actually finds a slightly better combination than the standard grid search and it is more efficient: it only checks about half the number of combinations than the standard grid search method. The savings in terms of time are also significant: on average it took around fifteen minutes to evaluate a combination, which includes the time to anonymize the data, to train a machine learning model on the anonymized data, and to apply the model on test data to measure TPR, FPR, and F1 score. Since our hybrid method checks just half of the combinations of the standard grid search, we can extrapolate that the hybrid model saves hours of tuning time compared to the standard grid search algorithm for this data set.

The hybrid method works well in this case because it first uses an initial grid search to locate promising areas, and then uses a greedy search to explore each promising area in more detail. This is a more efficient approach compared to the standard grid search which treats all cells (combinations) uniformly. As a result the standard grid search does not delve quickly around the most promising area (which is $k = 500, \epsilon = 0.1$ in this case).

5 CONCLUSION

This paper deals with the problem of anonymizing a data set to be used for analysis under the conditions that the privacy of the included data must be preserved as much as possible, it must withstand injection attacks, and maintain its utility. This is an important problem with a lot of applications in cyber security, network management, health care, and many other domains. In order to provide a solution to this problem, we proposed a self-adaptive and secure approach to anonymizing network trace data. Most existing anonymization methods for network data neither provide strong privacy protection, nor support automatic tuning of parameters. The proposed approach uses an anonymization with insensitive microaggregation method to provide strong privacy protection. A hybrid search approach is also introduced to automatically tune parameters. In addition, we proposed three methods for selecting the most appropriate features to anonymize. All three methods to select the best candidate features for anonymization also work equally well. In order

to evaluate our methods and our algorithms, we conducted several experiments. The experimental results validated our approach and verified that our techniques better preserve important properties in the data set and provide a strong privacy protection compared to other existing methods. In addition, when we applied our parameter tuning method and then anonymized our datasets, we observed results that were quite similar to those of a more expensive grid search approach.

5.1 Limitations and Future Work

There are two limitations in our approach. First, it assumes that class labels are present and data will be later classified. We also only used labeled data sets in our experiments. However, in practice, the data sets may not be labeled or only be partially labeled. It is possible though to directly apply our method on unlabeled data by just assuming that the entire data set belongs to one class. However, this may lead to a decrease of accuracy in subsequent analysis. In future work we plan to conduct experiments on unlabeled data sets. The second limitation is about the methods to select features for anonymization. So far all proposed methods only use properties of data. In the future we plan to consider data usage for selecting the most relevant features to anonymize. We also plan to further study various ways to help users fine tune the anonymization method based on specific analysis tasks. For example, if users want to detect scanning attacks where adversaries scan the network to gather information to launch attacks, it makes sense to preserve external source IP addresses in scanning traffic (where adversaries scan a range of IPs to collect information before attacking) because such addresses are used by adversaries to launch attacks.

REFERENCES

[1] Internet 2. 2014. Internet 2's Network Flow Data Privacy Policy. https://internet2.edu/security/routing-security/network-flow-data-privacy-policy/

[2] Rakesh Agrawal and Ramakrishnan Srikant. 2000. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 439–450.

[3] Ahmed Aleroud, Fan Yang, Sai Chaithanya Pallaprolu, Zhiyuan Chen, and George Karabatis. 2021. Anonymization of Network Traces Data through Condensation-based Differential Privacy. *Digital Threats: Research and Practice (DTRAP)* 2, 4 (2021), 1–23.

[4] Jasper Bongert. 2020. TcraceWrangler. https://www.tracewrangler.com/

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

[6] Martin Burkhart, Dominik Schatzmann, Brian Trammell, Elisa Boschi, and Bernhard Plattner. 2010. The role of network trace anonymization under attack. *ACM SIGCOMM Computer Communication Review* 40, 1 (2010), 5–11.

[7] Kai Cao, Yunchun Li, Hailong Yang, Jiqiang Tang, and Xiaoxiang Zou. 2015. Poster: An online prefix-preserving ip address anonymization algorithm for passive measurement systems. In *Security and Privacy in Communication Networks: 11th EAI International Conference, SecureComm 2015, Dallas, TX, USA, October 26-29, 2015, Proceedings 11*. Springer, 581–584.

[8] Shaveta Dandyan, Habib Louafi, and Samira Sadaoui. 2022. A Feistel Network-based Prefix-Preserving Anonymization Approach, Applied To Network Traces. In *2022 19th Annual International Conference on Privacy, Security & Trust (PST)*. IEEE, 1–11.

[9] Niels Van Dijkhuizen and Jeroen Van Der Ham. 2018. A survey of network traffic anonymisation techniques and implementations. *ACM Computing Surveys (CSUR)* 51, 3 (2018), 1–27.

[10] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*. Springer, 1–19.

[11] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[12] F. F. Prasser, K. Babioch Kohlmayer, I. Vujosevic, and R. Bild. 2021. Arx Data Anonymization Tool. https://arx.deidentifier.org/

[13] Roman Garnett. 2023. *Bayesian optimization*. Cambridge University Press.

[14] Francesco Gringoli. 2009. Tcpanon. http://netweb.ing.unibs.it/~ntw/tools/tcpanon/

[15] PACKET CLEARING HOUSE INC. 2018. Protected Repository for the Defense of Infrastructure Against Cyber Threats (PREDICT) Dataset Development and Hosting. (2018). https://apps.dtic.mil/sti/citations/AD1050331

[16] Christopher T Kenny, Shiro Kuriwaki, Cory McCartan, Evan TR Rosenman, Tyler Simko, and Kosuke Imai. 2021. The use of differential privacy for census data and its impact on redistricting: The case of the 2020 US Census. *Science advances* 7, 41 (2021), eabk3283.

[17] Dimitris Koukis, Spyros Antonatos, Demetres Antoniades, Evangelos P Markatos, and Panagiotis Trimintzios. 2006. A generic anonymization framework for network traffic. In *2006 IEEE International Conference on Communications*, Vol. 5. IEEE, 2302–2309.

[18] Ying-Dar Lin, Po-Ching Lin, Sheng-Hao Wang, I-Wei Chen, and Yuan-Cheng Lai. 2014. Pcaplib: A system of extracting, classifying, and anonymizing real packet traces. *IEEE Systems Journal* 10, 2 (2014), 520–531.

[19] Frank McSherry and Ratul Mahajan. 2010. Differentially-private network trace analysis. *ACM SIGCOMM Computer Communication Review* 40, 4 (2010), 123–134.

[20] Frank McSherry and Kunal Talwar. 2007. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE, 94–103.

[21] Meisam Mohammady, Lingyu Wang, Yuan Hong, Habib Louafi, Makan Pourzandi, and Mourad Debbabi. 2018. Preserving both privacy and utility in network trace anonymization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 459–474.

[22] Nour Moustafa and Jill Slay. 2015. The significant features of the UNSW-NB15 and the KDD99 data sets for network intrusion detection systems. In *2015 4th international workshop on building analysis datasets and gathering experience returns for security (BADGERS)*. IEEE, 25–31.

[23] Nour Moustafa and Jill Slay. 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 military communications and information systems conference (MilCIS)*. IEEE, 1–6.

[24] Jordi Soria-Comas, Josep Domingo-Ferrer, David Sánchez, and Sergio Martínez. 2014. Enhancing data utility in differential privacy via microaggregation-based k-anonymity. *The VLDB Journal* 23, 5 (2014), 771–794.

[25] Latanya Sweeney. 2002. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 05 (2002), 557–570.

[26] United States Code. 2022. U.S. Code - Unannotated Title 6. Domestic Security § 485 Information sharing (j)(2). . https://uscode.house.gov/view.xhtml?req=(title:6%20section:485%20edition:prelim).

[27] N. Vollmer. 2018. *Recital 26 Eu GDPR.* https://www.privacy-regulation.eu/en/recital-26-GDPR.htm

[28] Jun Xu, Jinliang Fan, Mostafa H Ammar, and Sue B Moon. 2002. Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In *10th IEEE International Conference on Network Protocols, 2002. Proceedings.* IEEE, 280–289.

[29] Li Yang and Abdallah Shami. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415 (2020), 295–316.