NetDPSyn: Synthesizing Network Traces under Differential Privacy

Danyu Sun
University of California, Irvine
Irvine, CA, USA
danyus2@uci.edu

Joann Qiongna Chen
San Diego State University
San Diego, CA, USA
jchen27@sdsu.edu

Chen Gong
University of Virginia
Charlottesville, VA, USA
fzv6en@virginia.edu

Tianhao Wang
University of Virginia
Charlottesville, VA, USA
tianhao@virginia.edu

Zhou Li
University of California, Irvine
Irvine, CA, USA
zhou.li@uci.edu

ABSTRACT

As the utilization of network traces for the network measurement research becomes increasingly prevalent, concerns regarding privacy leakage from network traces have garnered the public's attention. To safeguard network traces, researchers have proposed the *trace synthesis* that retains the essential properties of the raw data. However, previous works also show that synthesis traces with generative models are vulnerable under linkage attacks.

This paper introduces NetDPSyn, the first system to synthesize high-fidelity network traces under privacy guarantees. NetDPSyn is built with the Differential Privacy (DP) framework as its core, which is significantly different from prior works that apply DP when training the generative model. The experiments conducted on three flow and two packet datasets indicate that NetDPSyn achieves much better data utility in downstream tasks like anomaly detection. NetDPSyn is also 2.5 times faster than the other methods on average in data synthesis.

CCS CONCEPTS

• **Networks** → **Network measurement**; • **Security and privacy** → **Data anonymization and sanitization**.

KEYWORDS

differential privacy; synthetic data generation; network packets; network flows

ACM Reference Format:
Danyu Sun, Joann Qiongna Chen, Chen Gong, Tianhao Wang, and Zhou Li. 2024. NetDPSyn: Synthesizing Network Traces under Differential Privacy. In *Proceedings of the 2024 ACM Internet Measurement Conference (IMC '24), November 4–6, 2024, Madrid, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3646547.3689011

IMC '24, November 4–6, 2024, Madrid, Spain.

ACM ISBN 979-8-4007-0592-2/24/11
https://doi.org/10.1145/3646547.3689011

1 INTRODUCTION

Network traces are critical for guiding and designing many applications such as network telemetry and network-based anomaly detection. However, prominent privacy risks of sharing such data have to be considered, as the shared information might identify a participant or even leak sensitive attributes. The conventional approaches to contain information leakage from the published network traces mainly follow data redaction like IP anonymization [62]. However, such methods are vulnerable under inference attacks [26] and data reconstruction attacks [20, 31]. An alternative approach is to *synthesize* traces that capture the properties from the raw traces, gaining strong momentum as it addressed privacy regulations. Recently, Generative Adversarial Network (GAN) has been leveraged to synthesize network traces [18, 27, 34, 42, 54, 57, 61, 66]. However, there is no *guarantee* that the users' privacy behind the traces is well guarded, and Stadler et al. showed that the data synthesized under the generative models suffer from linkage attacks [58].

Differential Privacy (DP) [25], which incorporates calibrated noise to "hide" the existence of an individual's data, shows promise in providing the essential privacy guarantees against the aforementioned attacks. There has been early effort to add DP noises to the responses according to the *statistical queries* [49], but the traces were not released. Some recent works enhance the GAN-based trace synthesis with DP [27, 66], mainly through DP-SGD [6]. However, the data utility is significantly worsened even under a very relaxed privacy budget (see Section 3.1). *"Research on privacy-preserving network data sharing using rigorous approaches such as differential privacy is needed"* [37], but researchers also recognized that *"a critical gap remains identifying how these privacy-preserving technologies can be applied to networking problems"* [19].

In this work, we pursue a different direction from prior works [27, 66]: instead of synthesizing network records with generative models that is trained under DP, we try to capture the underlying distributions of the original data and synthesize network records from them *after* they are protected by DP. This design choice is driven by our insight that the underlying distributions are more critical to many downstream applications like anomaly detection, and by directly controlling these distributions under DP, we avoid the need for excessive noise addition when training the generative model. Specifically, we develop a new system NetDPSyn that extends a marginal-based synthesizer, PrivSyn [71], for the network settings.

We conducted a comprehensive study to evaluate the effectiveness of NetDPSyn, by comparing with 3 baseline methods on 5 datasets. Our initial result shows the data synthesized by NetDPSyn can achieve similar fidelity even as the raw data for tasks like flow classification. NetDPSyn is also much more efficient than the other baselines. Our code is available at https://github.com/DanyuSun/NetDPSyn.

2 BACKGROUND

2.1 Network Dataset and Privacy

Like previous works [42, 66], we consider *header* fields of network packets or flows as the target for data synthesis. Releasing the header fields is often sufficient to support many research scenarios like network-based anomaly detection [13], and addresses some privacy concerns (e.g., a payload might directly contain personal data). We use 5 public datasets that either contain packets or flows, as elaborated in Section 4.1. Below we describe their common fields.

- **Packet header.** It contains information about individual packets at layer 3 (IP layer) and layer 4 (transport layer). Specifically, it contains the source and destination IP addresses (`srcip` and `dstip`), source and destination ports (`srcport` and `dstport`), protocol type up to layer 4 (`proto`, e.g., TCP, UDP, and ICMP), timestamp of capture (`ts`), packet length (`pkt_len`), and other fields like checksum (`chksum`) and label given by the data collector (`label`).
- **Flow header.** A network flow aggregates the packets under IP 5-tuple ⟨`srcip`, `dstip`, `srcport`, `dstport`, `proto`⟩ [8]. The timestamp of the first packet (`ts`), the duration of flow (`td`), the number of packets (`pkt`), the number of bytes (`byt`), and flow label (`label`) are also included per flow.

Releasing the header without the payload still raises privacy concerns, and the main solutions include data anonymization and data synthesis. Data anonymization is often performed on IP addresses, e.g., with CryptoPan prefix-preserving anonymization [62]. Yet, a recent study showed such IP anonymization is still vulnerable if the institution associated with a prefix has sensitive Internet activities (e.g., sending email to a controversial organization) [37]. Data synthesis has finer-grained control over the privacy-utility tradeoff [66], and we aim to provide *provable* privacy protection for the synthesized network traces while maintaining their data utility.

2.2 Differential Privacy

Differential privacy (DP) [25] guarantees that the result of any computation on a dataset, such as a database query, remains essentially unchanged whether or not any single individual's data is included or excluded.

Definition 1. (*Differential Privacy [25]*) A randomized mechanism $\mathcal{A}$ satisfies (ε, δ)-differential privacy, if and only if, for any two neighboring datasets D and D', it holds that,

$$\Pr[\mathcal{A}(D) \in O] \leq e^{\varepsilon} \Pr[\mathcal{A}(D') \in O] + \delta, \tag{1}$$

where O represents the set of all conceivable outputs of the algorithm $\mathcal{A}$. The privacy budget ε and δ are both non-negative parameters that indicate the privacy loss in the data. A lower value of ε signifies enhanced privacy and a smaller δ corresponds to a decreased probability that the privacy protection assured by ε will be broken. The granularity of DP is dependent on how the neighboring dataset is defined. Typically, two datasets D, D' differing in only one record are considered neighboring, which can be regarded as *record-level* DP. For instance, this study defines the 'record' as a single log entry of a network packet or flow.

2.3 Related Works

Synthesis of network traces. Based on our literature review, recent works prefer to synthesize network traces with generative models. Among them, most of the works applied Generative Adversarial Networks (GANs) [18, 27, 34, 42, 54, 57, 61, 66], which trains *generator(s)* to map a noise vector to a sample and *discriminator(s)* to classify the sample as fake or real. NetShare is a prominent example [66], which has achieved high data fidelity on the header traces with a time-series generator.

Recently, a few works applied *diffusion* models to synthesize network traces [38, 56]. Take NetDiffus as an example [56]. It converts network traces into a sequence of 2-D images, iteratively adds noises in the forward pass, and trains a denoiser to recover the original network traces.

DP for dataset synthesis. DP has been extensively leveraged to construct synthetic data that can be shared under privacy guarantees [36]. Different types have been tested, including image [23, 32, 40, 41], tabular [14, 15, 46, 67, 70, 71], graph [68], time-series [29], trajectory [24, 60], text data [59, 69] and more.

NetDPSyn treats network traces as tabular data and follows marginal-based synthesis. Another direction is copula-based synthesis [30], which uses a copula function (e.g., Gaussian copula) to model the joint distribution and synthesize traces [7, 17, 30, 39]. We did preliminary experiments with Gaussian copula, but the result was unsatisfactory. Using a different copula function or adapting the Gaussian copula for the network datasets might be needed, and we leave it as a future work.

2.4 Threat Model

We follow the threat modeling of Houssiau et al. [35]: given a dataset synthesized from the original dataset, the attacker is motivated to conduct 3 types of attacks, including membership inference attack (MIA), attribute inference attack (AIA) and data reconstruction attack. MIA aims to determine whether a data point is included by or excluded from a synthetic dataset. AIA aims to infer sensitive attributes from the other released attributes. Data reconstruction aims to recover the entire original dataset. DP is supposed to deter all these inference attacks by modeling the worst-case scenario (i.e., strongest attacker) [22].

In Appendix G, we provide a preliminary evaluation of the effectiveness of the basic MIA method [65] on the synthesized network traces, suggesting DP is an effective defense.

3 NETDPSYN

3.1 Motivation and Workflow

We revisit the generative-model-based approaches for trace synthesis and argue that they cannot rigorously guard users' privacy. In

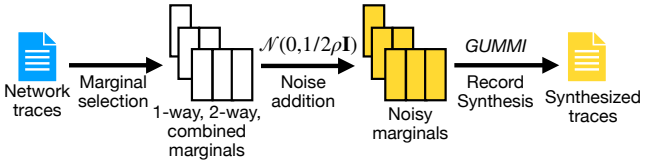

Figure 1: High-level workflow of NetDPSyn.

fact, Stadler et al. found the generative models like CTGAN [63] suffer from *linkage attacks*, which allows an attacker to infer the presence of a record in the original dataset with high confidence [58]. Though DP has been considered to harden the generative models [27, 66], we found they all choose DP-SGD [6], which clips each gradient and adds Gaussian noise to make the generative models fulfill (ϵ, δ)-DP. However, the data utility can be significantly worse. For example, Table 5 of [66] shows the Earth Mover's Distance (EMD) *is increased from 0.10 (without DP-SGD) to 0.35 (with DP-SGD), even under a very large* $\varepsilon = 24.24$. In fact, DP-SGD is known to add excessive noises due to that its privacy guarantee is proved on *each SGD step* [28].

Instead of synthesizing network records with generative models and modifying them to fulfill DP, we choose to capture the underlying distributions of the original data and synthesize network records from them after DP protects them. By doing so, we can directly control the privacy-utility tradeoff on the data distributions and avoid adding excessive noises. Figure 1 shows the general workflow of our approach.

To generate DP-protected correlations, we extend a prior work PrivSyn [71] which handles high-dimensional datasets by automatically selecting and constructing noisy marginal tables that capture the data distributions. It turns out to be more effective and efficient than the approaches based on probabilistic graphical model (e.g., Bayesian network [70] and Markov Random Fields [46–48]) as shown in our evaluation. We incorporate the *network domain knowledge* to adapt PrivSyn for network traces, by 1) developing a new binning technique to pre-process continuous attribute values; 2) augmenting the network traces with temporal features; 3) applying various protocol constraints for data consistency; 4) bootstrapping Gradually Update Method (GUM) of PrivSyn with an initialized dataset to scale up the synthesis process.

NetDPSyn consists of 3 main steps. First, it pre-processes the network traces with data binning and feature addition. Then, it selects 2-way marginals under an optimization procedure, publishes them by adding Gaussian noises, and post-processes the marginal tables to apply protocol-related restrictions for consistency. Finally, it synthesizes the network traces from the marginal tables. The workflow of NetDPSyn with pseudo-code is shown in Appendix B.

3.2 Pre-processing

Fields with large domain sizes could lead to excessive noise under DP mechanisms like Gaussian mechanisms, and PrivSyn combines *all* low-count values into a new value, but this approach would remove too many low-count values or merge values that are not relevant. We found this approach is particularly problematic for network traces, and we propose a finer-grained method to bin different network fields disparately based on their distribution and types.

As the first step, we use a *type-dependent* binning method on each attribute, and consider 5 field types. 1) IP (`srcip` and `dstip`): we bin the low-count IP addresses by the `/30` prefix. 2) Port (`srcport` and `dstport`): we keep a list of common ports under 1024 away from the binning process, and bin the higher port numbers by 10. 3) Categorical attributes (e.g., `proto` and `label`) with small domain size are not binned. 4) Integer and floating-point attributes (`pkt`, `byt` and `td`): they are binned under log transformation, i.e., $log(1+x)$, which results in much smaller number of bins than linear binning. 5) Timestamp (`ts`): we handle them with a new method detailed in the end of this subsection.

After the type-dependent binning, we perform another round of frequency-dependent binning, to further aggregate the bins with small frequencies. As frequency is related to the actual raw data, to control the information leakage introduced in this step, we apply DP before making the bin selection. In particular, for each attribute, we publish noisy 1-way marginals by applying Gaussian Mechanism (see "Adding Noise to Marginals" in Section 3.3) on the marginal cell count (e.g., $M_{\mathrm{d}}(v)$ in Table 4 of Appendix C), with a portion of privacy budget.

Capturing temporal pattern.

Inspired by [42], we create another field `tsdiff` (the differences between timestamps) from the `ts` field, which will be treated the same as other fields under data binning, marginal-table selection, etc. Instead of directly computing `tsdiff` on two adjacent time-ordered records, we group the records by an identifier, e.g., the IP 5-tuple, and compute the `tsdiff` in each group. We focus on the group-wise `tsdiff` as the activities of different groups are less likely to correlate. We use `tsdiff` to represent the temporal patterns because it reflects packet-arrival intervals, which are widely used in downstream applications, like autocorrelation [11].

3.3 Generating Noisy Marginal Tables

After the pre-processed 1-way marginal tables are generated, we can construct the 2-way marginal tables from them and publish their noisy versions using Gaussian Mechanism. The 2-way marginals capture the field correlations that are essential for high-fidelity record synthesis. However, publishing *all* 2-way marginal tables will introduce a large amount of DP noises, so we follow PrivSyn's DenseMarg algorithm to select useful 2-way marginals under low privacy budget. Though data owners can manually select marginals to be released, we argue that it is very difficult to select the right subset for the best privacy-utility tradeoff, so we take a data-driven approach. Examples of 1-way and 2-way marginal tables and their noisy versions are shown in Appendix C.

Marginal selection. DenseMarg formalizes the marginal selection problem as an optimization problem that balances dependency error (error caused by missing a marginal) and noise error (error caused by adding noises to a selected marginal) as below:

$$\text{minimize} \sum_{i=1}^{m} [\psi_i x_i + \phi_i (1 - x_i)] \text{ s.t. } x_i \in \{0, 1\} \tag{2}$$

where m is the number of all marginals ($m = d(d-1)/2$ for d fields), ψ_i is the noise error of selecting the i-th 2-way marginal, ϕ_i is the

dependency error of missing i, and $x_i \in \{0, 1\}$ indicates whether i is selected.

On top of noise error and dependency error, DenseMarg selects the optimal set of 2-way marginals with a greedy algorithm. Among the selected 2-way marginals, DenseMarg further merges the overlapping ones whose sizes are small, and derive the final set.

Adding noise to marginals. Gaussian noise is added to each selected marginal to satisfy DP. As proved in [71] (Theorem 6), a marginal M has a sensitivity of $\Delta_M = 1$, and the Gaussian mechanism dictates that the magnitude of the noise is contingent upon Δ_M. Consequently, the noisy marginal M is defined as, $\tilde{M} = M + \mathcal{N}(0, 1/2\rho\mathbf{I})$, where $\mathcal{N}(0, 1/2\rho\mathbf{I})$ denotes a multi-dimensional random variable following a normal distribution with a mean of 0 and a variance of $1/2\rho\mathbf{I}$.

Marginal post-processing. After the noisy marginals are published, every operation is post-processing, and no extra privacy budget will be consumed [25]. NetDPSyn conducts 3 steps to edit the published marginals to improve their utility. First, we project the invalid distribution into a valid one (i.e., no negative probabilities and their sum equals 1).

Second, when an attribute f is contained by multiple published marginals (e.g., $M_{f,a}$ and $M_{f,b}$), we use weighted average method [53] on the marginals to minimize the variance on f.

Third, we edit some entries in the marginal tables to make them consistent with protocol rules, e.g., `byt` has to be larger or equal to `pkt` (a packet should have at least one byte) and most of FTP packets should use TCP[1].

Privacy budget allocation. Given a privacy budget ρ, which is converted from (ε, δ) under Zero Concentrated DP [12], we allocate 0.1ρ, 0.1ρ, and 0.8ρ for data-dependent binning, marginal selection, and publishing noisy marginals.

3.4 Record Synthesis

After the noisy 1-way, 2-way, and combined marginals are derived, this step generates a synthesized dataset that has the same or similar marginals. We improve the Gradually Update Method (GUM) developed by PrivSyn for better efficiency, as this step takes most of execution time.

Specifically, we call our method *GUMMI (GUM with Marginal Initialization)*, which initializes a dataset D_s that contains marginals key to the downstream tasks, e.g., marginals that contain the *label* field as it is essential for flow/packet classification. As such, the correlations between the features (e.g., `dstport`) and the label are better preserved. We let the data owner specify the key attribute, and let GUMMI select the n^I noisy multi-way marginals that contain attribute, and orders them by *Pearson standard correlation coefficient* (high to low). Since the coefficient is computed on the noisy marginals, no privacy budget is consumed at this step. Next, GUM will be applied to iteratively update D_s to replace attributes or duplicate rows, so the the marginals from D_s are close enough to the released marginals.

Finally, each record needs to be decoded due to binning performed in the pre-processing stage. For the most binned fields, we uniformly sample a value within the bin. We also consider the network-related constraints to avoid sampling invalid values (e.g., port number should be less than 65536).

Regarding the timestamp field `ts`, we leverage the auxiliary field `tsdiff` generated during pre-processing (see Section 3.2) to synthesize its values. Specifically, we first cluster the encoded rows by their identifier, such as IP 5-tuple. Then, we sample within the `tsdiff` bin range under a Gaussian distribution (rounded to the nearest integer) and add the sampled value to the bin starts.

4 EVALUATION

In this section, we first describe the experimental setup, then compare the effectiveness and efficiency of NetDPSyn with other baseline methods in downstream tasks like packet/flow classification and data sketching. We report the results of attribute-wise measurement, ablation study, and privacy analysis in Appendix E, F, G.

4.1 Experimental Setup

Datasets and baseline models. We select 5 public datasets (UGR16, CIDDS, TON, CAIDA and DC), which are also used by our main baseline model NetShare [66][2]. These traces are diverse in the deployments, collection logic, and timescales. For baseline models, we consider NetShare, PGM [48] and PrivMRF [14]. We describe them in Appendix D.

Parameters of NetDPSyn. For most experiments, we set the privacy budget ε equal to 2.0 for NetDPSyn and the other baseline models, which is a common value used by other works (PrivSyn uses $\varepsilon \in [0.2, 2.0]$ [71]), providing moderate privacy guarantee. NetShare uses larger ε, from 24.24 to 10^8, and we argue that the privacy protection is significantly weakened. In Appendix F, we test other ε values.

For NetDPSyn, we set the maximum number of update iterations during record synthesis to 200. During marginal post-processing, we use τ to bound the probabilities of certain protocol combinations and we set its value to 0.1.

NetShare requires δ to be manually configured, we use the same value 10^{-5}. We also follow its "DP Pretrained-SAME" mode, which uses part of its data to pre-train a model and fine-tune the model with the remaining data. For PGM, we manually select all 2-way marginals that contain the label attribute of each dataset, which is expected to boost the accuracy on machine-learning based tasks.

Implementations and testing environment. We implement NetDPSyn in Python 3.11.4. All the experiments were run on a workstation with 20.04.1-Ubuntu, AMD 3970x CPU (32 cores) and 256GB memory.

4.2 Data Sketching

Many network applications leverage sketching to create a compact, efficient data structure for summarizing and analyzing network traffic in real-time. Like NetShare, we consider 4 common sketching algorithms, including Count-Min Sketch (CMS) [21], Count

[1]Though FTP is supposed to be on TCP, we found exceptions in our data: e.g., on the UGR16 dataset, there are 224 and 1,293 FTP packets (`dstport` is 21 or 22) using UDP. This could be caused by data collection errors or the abnormal behaviors of clients. So instead of removing the entries that violate protocol rules, we assign a probability threshold τ.

[2]We did not include the Cyber Attack (CA) dataset [3] like NetShare, as the original data does not have a label and we did not find another dataset that includes the label attribute.

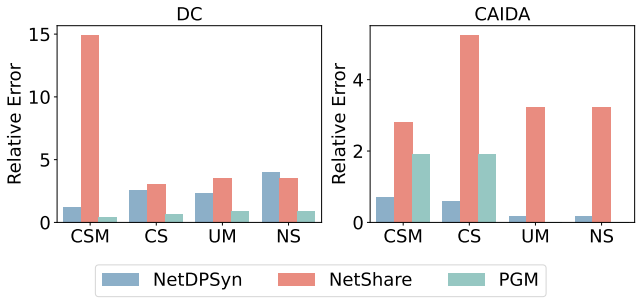

Figure 2: Relative error of various sketch algorithms. The lower the better.

Sketch (CS) [21], Universal Monitoring (UM) [44], and NitroSketch (NS) [43]. The threshold for heavy hitters is set to 0.1%.

We compute relative error for heavy hitter count estimation between synthesized and raw data. Assuming the errors for synthesized and raw data are err_{syn} and err_{raw}, the relative error is $|\frac{err_{syn}-err_{raw}}{err_{raw}}|$. We use two packet datasets, CAIDA and DC, for this task and compute heavy hitter counts on CAIDA's `srcip` and DC's `dstip`. As the sketching algorithms have randomness, we run each sketch 10 times. In Figure 2, we show the results of 4 sketching methods on the datasets, and NetShare is significantly worse than the other methods, except for NitroSketch on DC. In fact, NetShare performs particularly worse for simpler sketching algorithms, like CSM on DC (12x relative error compared to NetDPSyn) and CS on CAIDA (9x relative error compared to NetDPSyn). NetDPSyn outperforms PGM in CAIDA when CSM and CS are used. For PrivMRF, we found it cannot run on DC and CAIDA as it exceeds our machine's memory.

4.3 Machine-learning Tasks

Next, we consider machine-learning tasks, which highly depend on feature engineering, to assess whether the attribute correlations are preserved.

Classification on flows. Anomaly detection is a key use case on network flows. We consider 3 datasets, including TON, UGR16, and CIDDS, for this task. For TON, we use its `type` attribute as the classification label, which describes the attack type (10 categorical values including "normal", "ddos", etc.). For UGR16 and CIDDS, we use their binary `label` attribute (benign or malicious) as the classification label. All attributes except the classification label are used as features. We randomly split each data into 80% for training and 20% for testing[3] and we measured classification accuracy, which is defined as $\frac{TP+TN}{TP+TN+FP+FN}$, where TP, TN, FP and FN are true positives, true negatives, false positives and false negatives. We implemented five common models: Decision Tree (DT), Logistic Regression (LR), Random Forest (RF), Gradient Boosting (GB), and Multi-layer Perceptron (MLP).

[3]NetShare claimed to split the data by time orders in the paper. We initially followed the same data split method for TON, but the classification accuracy even on raw data is low. We found most of the simulated attacks in TON happen at the end of the period, so the training data will not contain sufficient attack records if splitting by time orders.

Table 1: Spearman's rank correlation of prediction algorithms on TON, CIDDS and UGR16. The higher the better. "N/A" for PrivMRF at CIDDS and UGR16 as memory is exceeded.

	NetDPSyn	NetShare	PGM	PrivMRF
TON	0.90	-0.90	0.70	-0.50
CIDDS	0.90	-0.30	0.70	N/A
UGR16	0.45	0.36	0.36	N/A

Figure 3 compares the classification accuracy of the 3 datasets. The performance gap on TON is most prominent, as the data generated by PGM and NetDPSyn lead to close accuracy as the raw data (e.g., 0.987 for Raw data, 0.889 for NetDPSyn, 0.886 for PGM with DT), but the data from NetShare lead to significantly lower accuracy (e.g., 0.235 with DT). The accuracy with LR is low for all models, mainly due to the simplicity of LR. For UGR16 and CIDDS, the data generated by most methods lead to very high accuracy, which is close to the accuracy on raw data. This is mainly because the classification label is binary and the data is highly imbalanced, so achieving high accuracy is much easier (e.g., even predicting every row as benign on UGR16 achieves 0.997 accuracy). Still, the data from NetShare lead to 0.1 to 0.2 lower accuracy on UGR16.

Instead of always achieving high accuracy, it is more important that a classification model achieves similar accuracy on raw and synthesized datasets for fidelity, even if the accuracy is low in both cases. As such, we compare the rankings of the 5 models when they are trained/tested on raw/synthetic data. Like NetShare, we compute Spearman's rank correlation coefficient between the raw/synthetic data. Table 1 shows that NetDPSyn achieves the highest coefficient.

We also found NetDPSyn achieves similar or even better data utility than NetShare *without DP*. For example, the rank correlations are 0.70 TON and 0.90 CIDDS for NetShare (Table 3 [66]), while 0.90 and 0.90 for NetDPSyn (Table 1).

Anomaly detection on packets. For packet datasets, we leveraged NetML [64], an open-source library to generate flow representations and leverage its the default one-class support vector machine (OCSVM) to detect abnormal packets. Like NetShare, we choose 5 modes of NetML: IAT, SIZE, IAT_SIZE, STATS, SAMP-NUMP (SN) and SAMP-SIZE (SS). As an example, STATS contains 10 statistical features from a flow, like flow duration, number of packets per second, etc.

For each round of running OCSVM, we obtain an anomaly ratio for the raw data (ano_{raw}) and one for the synthesized data (ano_{syn}). We compute the relative error as $|\frac{ano_{syn}-ano_{raw}}{ano_{raw}}|$. As NetML only accepts flows with at least two packets, a subset of packets can be used. Figure 4 shows the results. In addition to PrivMRF being unable to run on the two packet datasets, we found that PGM also encounters "NaN" on CAIDA. This is because in the dataset generated by PGM, only a few number of flows contain two packets. We found NetDPSyn has comparable results as NetShare except SS. PGM has a very high relative error for SIZE and SS.

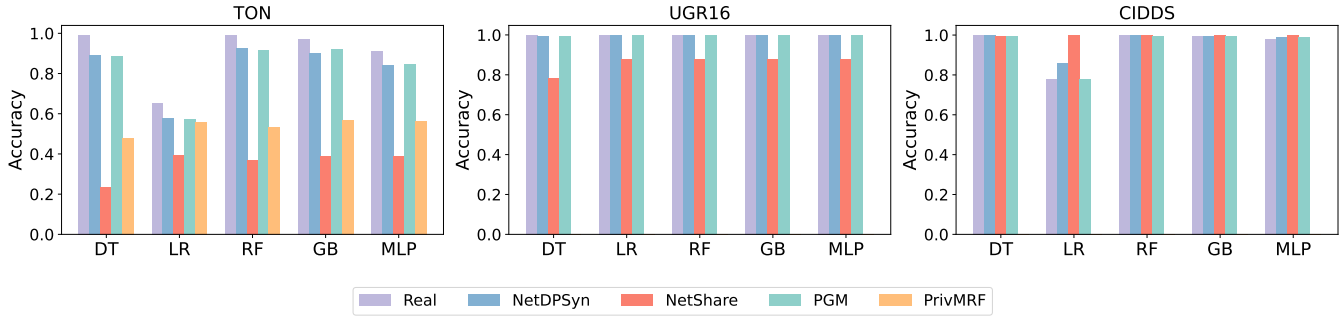

Figure 3: Classification accuracy of three flow datasets. The higher the better.

Table 2: Rank correlation of NetML model for packet anomaly detection. The higher the better.

	NetDPSyn	NetShare	PGM	PrivMRF
CAIDA	-0.48	-0.82	N/A	N/A
DC	0.26	-0.65	-0.26	N/A

Table 3: Running Time of each method in minutes.

	NetDPSyn	NetShare	PGM	PrivMRF
TON	10	27	70	240
CIDDS	20	100	55	N/A
UGR16	40	94	55	N/A
CAIDA	35	30	54	N/A
DC	20	100	24	N/A

We also compute the Spearman's rank correlation coefficient between the raw/synthetic data, and the result is shown in Table 2. NetDPSyn performs best.

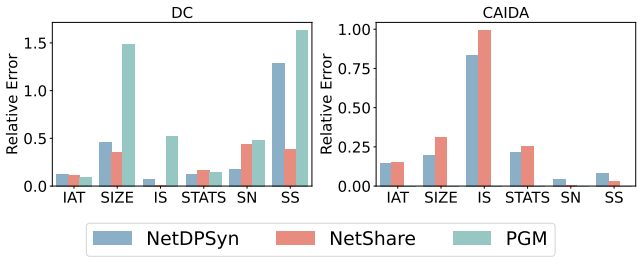

Figure 4: NetML results on two packet datasets. The lower the better.

4.4 Efficiency

We compare the efficiency of different methods by measuring their running time, as shown in Table 3. NetDPSyn is much faster than NetShare and PGM in most cases, e.g., it is 5x and 2x faster than Netshare and PGM in processing CIDDS. PrivMRF is particularly slow, e.g., taking 240 minutes to synthesize TON, and it even exceeds the memory limit when processing datasets larger than TON. This is because PrivMRF selects too many marginals.

5 CONCLUSION AND FUTURE WORKS

This paper introduces NetDPSyn, a method for synthesizing high-fidelity network traces with Differential Privacy. We take a drastically different path from prior works based on generative models like NetShare. Specifically, NetDPSyn uses DP as the core to capture the underlying distribution of the high-dimensional network traces and synthesize records from them under DP's post-processing property. Our initial results are promising, showing significantly better data fidelity and efficiency than baseline methods like NetShare, PGM and PrivMRF, under the same DP privacy budget.

Limitations and future works. We consider our work as the first step, and we acknowledge a few limitations as listed below. 1) We model the temporal patterns with packet-arrival intervals, it is relatively coarse representation. NetShare might outperform NetDPSyn in downstream tasks that require complex temporal modeling. Combining NetDPSyn with recurrent neural networks (RNN) might lead to better results in this aspect. 2) NetDPSyn does not cover all types of network environment and data types (e.g., payload data). 3) NetDPSyn performs well when the number of attributes is relatively small (e.g. all tested datasets have no more than 15 attributes). High-dimensional data might lead to computational and memory inefficiencies, and we can integrate dimensionality reduction techniques to address this issue. 4) We did not evaluate advanced downstream tasks like graph-based anomaly detection [52], which could be an interesting problem to study in the future.

ACKNOWLEDGEMENT

We thank our reviewers and shepherd for the valuable suggestions. We thank Rui Zhao for his contribution in the initial system implementation. This project is supported under NSF CNS-2220434 and CNS-2220433.

REFERENCES

[1] 2020. PGM code repo. https://github.com/ryan112358/private-pgm.
[2] 2022. The CAIDA UCSD Anonymized Internet Traces. https://www.caida.org/catalog/datasets/passive_dataset.
[3] 2022. Capture files from Mid-Atlantic CCDC. https://www.netresec.com/?page=MACCDC.
[4] 2023. NetShare code repo. https://github.com/netsharecmu/NetShare.
[5] 2023. PrivMRF code repo. https://github.com/caicre/PrivMRF.
[6] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[7] Hassan Jameel Asghar, Ming Ding, Thierry Rakotoarivelo, Sirine Mrabet, and Dali Kaafar. 2020. Differentially private release of datasets using Gaussian copula. *Journal of Privacy and Confidentiality* 10, 2 (2020).

[8] M Bagnulo, P Matthews, and I van Beijnum. 2011. *Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers*. Technical Report.

[9] Theophilus Benson, Aditya Akella, and David A. Maltz. 2010. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference, IMC 2010, Melbourne, Australia - November 1-3, 2010*, Mark Allman (Ed.). ACM, 267–280.

[10] Theophilus Benson, Aditya Akella, and David A Maltz. 2010. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. 267–280.

[11] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

[12] Mark Bun and Thomas Steinke. 2016. Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds. In *Theory of Cryptography - 14th International Conference, TCC 2016-B (Lecture Notes in Computer Science, Vol. 9985)*, Martin Hirt and Adam D. Smith (Eds.). 635–658.

[13] Philipp Bönninghausen. 2024. Intrusion Detection Datasets: An overview of datasets suitable for IDS research. https://fkie-cad.github.io/intrusion-detection-datasets/content/all_datasets/.

[14] Kuntai Cai, Xiaoyu Lei, Jianxin Wei, and Xiaokui Xiao. 2021. Data Synthesis via Differentially Private Markov Random Fields. *Proc. VLDB Endow.* 14, 11 (jul 2021), 2190–2202.

[15] Kuntai Cai, Xiaokui Xiao, and Graham Cormode. 2023. PrivLava: Synthesizing Relational Data with Foreign Keys under Differential Privacy. *Proc. ACM Manag. Data* 1, 2, Article 142 (jun 2023).

[16] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. 2022. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1897–1914.

[17] Thee Chanyaswad, Changchang Liu, and Prateek Mittal. 2019. RON-Gauss: Enhancing Utility in Non-Interactive Private Data Release. *Proceedings on Privacy Enhancing Technologies* 1 (2019), 26–46.

[18] Adriel Cheng. 2019. PAC-GAN: Packet generation of network traffic using generative adversarial networks. In *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE, 0728–0734.

[19] KC Claffy, David Clark, John Heidemann, Fabian Bustamante, Mattijs Jonker, Aaron Schulman, and Ellen Zegura. 2021. Workshop on Overcoming Measurement Barriers to Internet Research (WOMBIR 2021) Final Report. *ACM SIGCOMM Computer Communication Review* 51, 3 (2021), 33–40.

[20] Aloni Cohen and Kobbi Nissim. 2018. Linear program reconstruction in practice. *arXiv preprint arXiv:1810.05692* (2018).

[21] Graham Cormode and Shan Muthukrishnan. 2005. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms* 55, 1 (2005), 58–75.

[22] Emiliano De Cristofaro. 2024. Synthetic Data: Methods, Use Cases, and Risks. *IEEE Security & Privacy* (2024).

[23] Tim Dockhorn, Tianshi Cao, Arash Vahdat, and Karsten Kreis. 2023. Differentially Private Diffusion Models. *Transactions on Machine Learning Research* (2023). https://openreview.net/forum?id=ZPpQk7FJXF

[24] Yuntao Du, Yujia Hu, Zhikun Zhang, Ziquan Fang, Lu Chen, Baihua Zheng, and Yunjun Gao. 2023. LDPTrace: Locally Differentially Private Trajectory Synthesis. *arXiv preprint arXiv:2302.06180* (2023).

[25] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference,*. Springer, 265–284.

[26] Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. 2017. Exposed! a survey of attacks on private data. *Annual Review of Statistics and Its Application* 4 (2017), 61–84.

[27] Liyue Fan and Akarsh Pokkunuru. 2021. DPNeT: Differentially private network traffic synthesis with generative adversarial networks. In *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 3–21.

[28] Vitaly Feldman, Ilya Mironov, Kunal Talwar, and Abhradeep Thakurta. 2018. Privacy amplification by iteration. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 521–532.

[29] Lorenzo Frigerio, Anderson Santana de Oliveira, Laurent Gomez, and Patrick Duverger. 2019. Differentially Private Generative Adversarial Networks for Time Series, Continuous, and Discrete Open Data. In *ICT Systems Security and Privacy Protection*, Vol. 562. Springer, 151–164.

[30] Sébastien Gambs, Frédéric Ladouceur, Antoine Laurent, and Alexandre Roy-Gaumond. 2021. Growing synthetic data through differentially-private vine copulas. *Proc. Priv. Enhancing Technol.* 2021, 3 (2021), 122–141.

[31] Simson Garfinkel, John M Abowd, and Christian Martindale. 2019. Understanding database reconstruction attacks on public data. *Commun. ACM* 62, 3 (2019), 46–53.

[32] Sahra Ghalebikesabi, Leonard Berrada, Sven Gowal, Ira Ktena, Robert Stanforth, Jamie Hayes, Soham De, Samuel L. Smith, Olivia Wiles, and Borja Balle. 2023. Differentially Private Diffusion Models Generate Useful Synthetic Images. *CoRR* abs/2302.13861 (2023).

[33] Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. 2021. User-level differentially private learning via correlated sampling. *Advances in Neural Information Processing Systems* 34 (2021), 20172–20184.

[34] Luchao Han, Yiqiang Sheng, and Xuewen Zeng. 2019. A packet-length-adjustable attention model based on bytes embedding using flow-wgan for smart cybersecurity. *IEEE Access* 7 (2019), 82913–82926.

[35] Florimond Houssiau, James Jordon, Samuel N. Cohen, Owen Daniel, Andrew Elliott, James Geddes, Callum Mole, Camila Rangel-Smith, and Lukasz Szpruch. 2022. TAPAS: a Toolbox for Adversarial Privacy Auditing of Synthetic Data. arXiv:2211.06550 [cs.CR] https://arxiv.org/abs/2211.06550

[36] Yuzheng Hu, Fan Wu, Qinbin Li, Yunhui Long, Gonzalo Munilla Garrido, Chang Ge, Bolin Ding, David Forsyth, Bo Li, and Dawn Song. 2023. SoK: Privacy-Preserving Data Synthesis. *arXiv preprint arXiv:2307.02106* (2023).

[37] Basileal Imana, Aleksandra Korolova, and John Heidemann. 2021. Institutional privacy risks in sharing DNS data. In *Proceedings of the Applied Networking Research Workshop*. 69–75.

[38] Xi Jiang, Shinan Liu, Aaron Gember-Jacobson, Arjun Nitin Bhagoji, Paul Schmitt, Francesco Bronzino, and Nick Feamster. 2023. NetDiffusion: Network Data Augmentation Through Protocol-Constrained Traffic Generation. *arXiv preprint arXiv:2310.08543* (2023).

[39] Haoran Li, Li Xiong, and Xiaoqian Jiang. 2014. Differentially private synthesization of multi-dimensional data using copula functions. In *Advances in database technology: proceedings. International conference on extending database technology*, Vol. 2014. NIH Public Access, 475.

[40] Kecen Li, Chen Gong, Zhixiang Li, Yuzhong Zhao, Xinwen Hou, and Tianhao Wang. 2023. Meticulously Selecting 1% of the Dataset for Pretraining! Generating Differentially Private Images Data with Semantics Query. arXiv:2311.12850 [cs.CV]

[41] Seng Pei Liew, Tsubasa Takahashi, and Michihiko Ueno. 2022. PEARL: Data Synthesis via Private Embeddings and Adversarial Reconstruction Learning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=M6M8BEmd6dq

[42] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. 2020. Using gans for sharing networked time series data: Challenges, initial promise, and open questions. In *Proceedings of the ACM Internet Measurement Conference*. 464–483.

[43] Zaoxing Liu, Ran Ben-Basat, Gil Einziger, Yaron Kassner, Vladimir Braverman, Roy Friedman, and Vyas Sekar. 2019. Nitrosketch: Robust and general sketch-based monitoring in software switches. In *Proceedings of the ACM Special Interest Group on Data Communication*. 334–350.

[44] Zaoxing Liu, Antonis Manousis, Gregory Vorsanger, Vyas Sekar, and Vladimir Braverman. 2016. One sketch to rule them all: Rethinking network flow monitoring with univmon. In *Proceedings of the 2016 ACM SIGCOMM Conference*. 101–114.

[45] Gabriel Maciá-Fernández, José Camacho, Roberto Magán-Carrión, Pedro García-Teodoro, and Roberto Therón. 2018. UGR'16: A new dataset for the evaluation of cyclostationarity-based network IDSs. *Comput. Secur.* 73 (2018), 411–424.

[46] Ryan McKenna, Brett Mullins, Daniel Sheldon, and Gerome Miklau. 2022. AIM: An Adaptive and Iterative Mechanism for Differentially Private Synthetic Data. *Proc. VLDB Endow.* 15, 11 (2022), 2599–2612.

[47] Ryan McKenna, Siddhant Pradhan, Daniel R Sheldon, and Gerome Miklau. 2021. Relaxed marginal consistency for differentially private query answering. *Advances in Neural Information Processing Systems* 34 (2021), 20696–20707.

[48] Ryan McKenna, Daniel Sheldon, and Gerome Miklau. 2019. Graphical-model based estimation and inference for differential privacy. In *Proceedings of the 36th International Conference on Machine Learning, ICML (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 4435–4444.

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

[50] Matthieu Meeus, Florent Guepin, Ana-Maria Creţu, and Yves-Alexandre de Montjoye. 2023. Achilles' heels: vulnerable record identification in synthetic data publishing. In *European Symposium on Research in Computer Security*. Springer, 380–399.

[51] Nour Moustafa. 2021. A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustainable Cities and Society* 72 (2021), 102994.

[52] Alina Oprea, Zhou Li, Ting-Fang Yen, Sang H Chin, and Sumayah Alrwais. 2015. Detection of early-stage enterprise infection by mining large-scale log data. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 45–56.

[53] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2014. Priview: practical differentially private release of marginal contingency tables. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1435–1446.

[54] Markus Ring, Daniel Schlör, Dieter Landes, and Andreas Hotho. 2019. Flow-based network traffic generation using generative adversarial networks. *Computers & Security* 82 (2019), 156–172.

[55] Markus Ring, Sarah Wunderlich, Dominik Grüdl, Dieter Landes, and Andreas Hotho. 2017. Creation of flow-based data sets for intrusion detection. *Journal of Information Warfare* 16, 4 (2017), 41–54.
[56] Nirhoshan Sivaroopan, Dumindu Bandara, Chamara Madarasingha, Guilluame Jourjon, Anura Jayasumana, and Kanchana Thilakarathna. 2023. NetDiffus: Network Traffic Generation by Diffusion Models through Time-Series Imaging. *arXiv preprint arXiv:2310.04429* (2023).
[57] Nirhoshan Sivaroopan, Chamara Madarasingha, Shashika Muramudalige, Guillaume Jourjon, Anura Jayasumana, and Kanchana Thilakarathna. 2023. SyNIG: Synthetic Network Traffic Generation through Time Series Imaging. In *2023 IEEE 48th Conference on Local Computer Networks (LCN)*. IEEE, 1–9.
[58] Theresa Stadler, Bristena Oprisanu, and Carmela Troncoso. 2022. Synthetic data–anonymisation groundhog day. In *31st USENIX Security Symposium (USENIX Security 22)*. 1451–1468.
[59] Xinyu Tang, Richard Shin, Huseyin A. Inan, Andre Manoel, Fatemehsadat Mireshghallah, Zinan Lin, Sivakanth Gopi, Janardhan Kulkarni, and Robert Sim. 2023. Privacy-Preserving In-Context Learning with Differentially Private Few-Shot Generation. arXiv:2309.11765 [cs.LG]
[60] Haiming Wang, Zhikun Zhang, Tianhao Wang, Shibo He, Michael Backes, Jiming Chen, and Yang Zhang. 2023. PrivTrace: Differentially Private Trajectory Synthesis by Adaptive Markov Model. In *USENIX Security Symposium 2023*.
[61] Pan Wang, Shuhang Li, Feng Ye, Zixuan Wang, and Moxuan Zhang. 2020. PacketCGAN: Exploratory study of class imbalance for encrypted traffic classification using CGAN. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 1–7.
[62] 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.
[63] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling tabular data using conditional gan. *Advances in neural information processing systems* 32 (2019).
[64] Kun Yang, Samory Kpotufe, and Nick Feamster. 2020. A comparative study of network traffic representations for novelty detection. *arXiv preprint arXiv:2006.16993* (2020).
[65] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*. IEEE, 268–282.
[66] Yucheng Yin, Zinan Lin, Minhao Jin, Giulia Fanti, and Vyas Sekar. 2022. Practical gan-based synthetic ip header trace generation using netshare. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 458–472.
[67] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. 2019. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In *International Conference on Learning Representations*.
[68] Quan Yuan, Zhikun Zhang, Linkang Du, Min Chen, Peng Cheng, and Mingyang Sun. 2023. PrivGraph: Differentially Private Graph Data Publication by Exploiting Community Information. In *32nd USENIX Security Symposium, USENIX Security 2023*. USENIX Association, 3241–3258.
[69] Xiang Yue, Huseyin Inan, Xuechen Li, Girish Kumar, Julia McAnallen, Hoda Shajari, Huan Sun, David Levitan, and Robert Sim. 2023. Synthetic Text Generation with Differential Privacy: A Simple and Practical Recipe. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1321–1342.
[70] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2014. PrivBayes: private data release via bayesian networks. In *International Conference on Management of Data, SIGMOD*. ACM, 1423–1434.
[71] Zhikun Zhang, Tianhao Wang, Ninghui Li, Jean Honorio, Michael Backes, Shibo He, Jiming Chen, and Yang Zhang. 2021. {PrivSyn}: Differentially Private Data Synthesis. In *30th USENIX Security Symposium (USENIX Security 21)*. 929–946.

A ETHICS

We evaluated NetDPSyn with the common downstream tasks like machine-learning-based classification and data sketching, using the same set of public datasets as NetShare. We do not foresee any ethical issue in this aspect. A potential concern is about the inference attack we conducted to evaluate the privacy leakage of the raw and synthesized data. We follow a similar attack procedure as Chanyaswad et al. [17], focusing on assessing system vulnerabilities rather than attempting to de-anonymize any data. Only aggregated statistics like attack accuracy is derived. It is important to note that our intent was to underline the importance of robust privacy protections, not to compromise any individual's anonymity.

B WORKFLOW OF NETDPSYN

In Algorithm 1, we show workflow of NetDPSyn in pseudo-code, consisting of pre-processing, marginal selection, noise addition, and record synthesis.

Algorithm 1 Pseudo-code of NetDPSyn.

Require: Private dataset Do, privacy budget ρ
Ensure: Synthetic dataset Ds

```
Binning each attribute using type-dependent binning method
Add the auxiliary temporal attribute
Publish 1-way marginals using GM with ρ1 = 0.1ρ
Binning each attribute with frequency-dependant method
Select 2-way marginals with ρ2 = 0.1ρ
Combine marginals with small sizes
Publish combined marginals using Gaussian Mechanism with ρ3 = 0.8ρ
Make noisy marginals consistent on the sum of cell values, shared attributes, and protocol rules
Construct encoded dataset De using GUMMI from an initialized dataset Ds
Decode De by value sampling within bins
Reconstruct the timestamp attribute with the auxiliary temporal attribute
```

C EXAMPLES OF MARGINAL TABLES

In Table 4, we show examples of 1-way and 2-way marginal tables and their noisy versions.

v	$M_{\text{d}}(v)$
⟨53, *⟩	82828
⟨80, *⟩	68748
⟨15600, *⟩	27255

(a) 1-way marginal for `dstport`.

v	$M_{\text{t}}(v)$
⟨∗, normal⟩	166494
⟨∗, injection⟩	15951

(b) 1-way marginal for `type`.

v	$M_{\text{dt}}(v)$
⟨53, normal⟩	74547.08
⟨53, injection⟩	554.71
⟨80, normal⟩	12297.88
⟨80, injection⟩	15396.66
⟨15600, normal⟩	27247.02
⟨15600, injection⟩	20.09

(c) 2-way noisy marginal before marginal post-processing.

v	$\tilde{M}_{\text{dt}}(v)$
⟨53, normal⟩	74566
⟨53, injection⟩	558
⟨80, normal⟩	12308
⟨80, injection⟩	15364
⟨15600, normal⟩	27255
⟨15600, injection⟩	0

(d) Actual 2-way marginal.

Table 4: Marginal tables for `dstport` and `type` computed on TON dataset. Due to space limit, only the first few rows are shown.

D DATASETS AND BASELINE MODELS

We obtained copies of datasets from the authors of NetShare, which are subsets of original datasets. In Table 5, we describe the basic statistics of each dataset.

- **URG16** [45]: This dataset comprises network traffic collected from NetFlow v9 collectors within a Spanish ISP's network, including various attacks. The specific data was gathered during the third week of March 2016.
- **CIDDS** [55]: The dataset replicates a small business environment featuring various clients and servers (such as email and web services) into which malicious traffic has been intentionally introduced. Each NetFlow entry is meticulously recorded, classified as benign or an attack, and categorized by attack type, including DoS, brute force, and port scans.
- **TON_IoT (TON)** [51]: TON is a collection representing telemetry from IoT sensors. Our evaluations focus on a subset of this dataset named "Train_Test_datasets." Cyber-attacks, such as backdoor, DDoS, and injection, are simulated.
- **CAIDA** [2]: This dataset has anonymized data gathered from high-speed monitors located on a commercial backbone network.
- **Data Center (DC)** [9]: It is a collection of packet captures from the "UNI1" data center, which is used by [10].

Table 5: Summary of datasets used in our experiments. Domain is computed by summing the domain sizes from all attributes.

Dataset	Records	Attributes	Domain	Label	Type
TON	295,497	11	$2 \cdot 10^6$	type	flow
UGR16	1,000,000	10	$4 \cdot 10^6$	type	flow
CIDDS	1,000,000	11	$6 \cdot 10^6$	type	flow
CAIDA	1,000,000	15	$1 \cdot 10^7$	flag	packet
DC	1,000,000	15	$1 \cdot 10^7$	flag	packet

Regarding baseline models to compare with NetDPSyn, we mainly use the GAN-based NetShare. We also consider PGM [48] and PrivMRF [14], which are two other marginal-based synthesis approaches. These methods are briefly described below.

- **GAN-based NetShare [66].** It uses a time-series generator to generate packets' metadata and their measurements, and then uses one discriminator to differentiate the packet time series and another auxiliary discriminator to discriminate only on metadata. To better capture the header field correlation spanning multiple packets or epochs, NetShare splits the network data by flows and uses the time-series GAN to synthesize new flows in parallel. To fairly compare with NetDPSyn and other baseline models, we use its DP version that applies DP-SGD when training the time-series GAN. We use its code from [4].
- **PGM [48].** This approach concurrently selects marginal distributions and establishes the Bayesian network's structure. It does so by iteratively optimizing the information gain using the exponential mechanism. Following this process, synthetic data is generated by sampling from the joint distribution derived from the established topology of the Bayesian network. The major limitation of PGM is that the operator needs to manually provide a list of marginals for PGM to synthesize from. We use its code from [1].
- **PrivMRF [14].** This method addresses the limitations of PGM by using a DP algorithm to automatically select a set of low-dimensional marginals. Selected marginals are used to construct a Markov random field (MRF), which models dependencies among attributes in the input data. The MRF then serves as the basis for generating synthetic data. We use its code from [5].

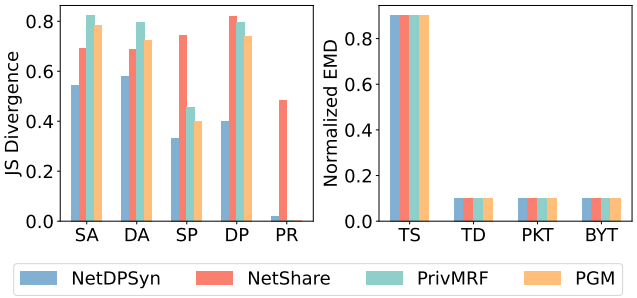

Figure 5: TON (NetFlow) JSD and EMD. The lower the better.

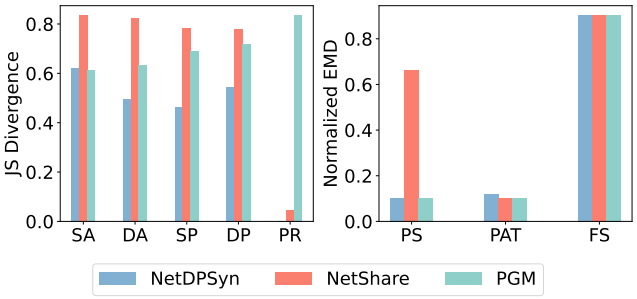

Figure 6: CAIDA (Packet) JSD and EMD. The lower the better.

E ATTRIBUTE-WISE MEASUREMENT

We measure the value distribution of individual attributes after synthesis. Like NetShare, for categorical attributes, we use Jensen-Shannon divergence (JSD) to measure the distance between the synthesized attribute and the raw attribute. For continuous attributes, we use Earth Mover's Distance (EMD). Because different attributes have vastly different EMD ranges, we normalize the EMDs to $[0.1, 0.9]$ for better figure readability.

Regarding categorical attributes, for both network flows and packets, we compute 5 metrics, including SA (relative frequency of `srcip` ranking in a descending way), DA (same metric for `dstip`), SP (port number distribution of `srcport`, ranging from 0 to 65535), DP (same metric for `dstport`), and PR (relative frequency of `proto`).

Regarding continuous attributes, the metrics are different for network flows and packets. For flows, we directly compute EMD on `ts`, `td`, `pkt`, and `byt`. `ts` and `td` are in milliseconds. For packets, we compute PS (Packet Size in bytes, same as `pkt_len`), PAT (Packet Arrival Time in milliseconds, same as `ts`), and FS (Flow Size, or the number of packets under an IP 5-tuple). The explanations of the attributes are in Section 2.1.

	DT		RF	
ε	NetDPSyn	NetShare	NetDPSyn	NetShare
4.0	0.910	0.213	0.932	0.368
16.0	0.941	0.235	0.954	0.389
32.0	0.943	0.257	0.955	0.413
64.0	0.946	0.258	0.955	0.427
10^3	0.947	0.260	0.956	0.423
10^{10}	0.948	0.389	0.957	0.580

Table 6: Comparison of TON (NetFlow) accuracy between NetDPSyn and NetShare with a large range of ε.

	DT		RF	
ε	NetDPSyn	NetShare	NetDPSyn	NetShare
4.0	0.976	0.779	0.986	0.870
16.0	0.978	0.781	0.987	0.874
32.0	0.980	0.781	0.988	0.875
64.0	0.983	0.782	0.988	0.878
10^3	0.986	0.783	0.990	0.878
10^{10}	0.989	0.783	0.992	0.879

Table 7: Comparison of UGR16 (NetFlow) accuracy between NetDPSyn and NetShare with a large range of ε.

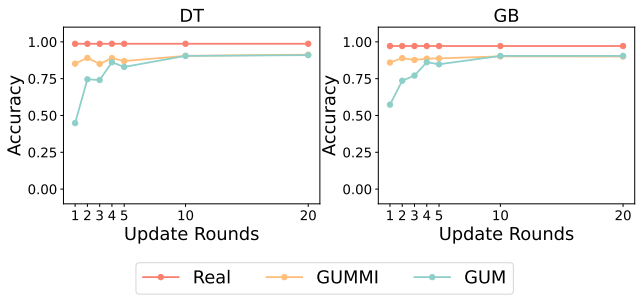

Figure 8: Comparison between GUMMI and GUM in update iterations from 1 to 20. The classification accuracies of DT and GB on TON are shown.

In Figure 5 and Figure 6, we show the results on one network flow dataset (TON) and one packet dataset (CAIDA). For the categorical metrics in TON, NetDPSyn is consistently better than the other methods, with 30%-45% lower JSD. NetShare performs notably worse in PR, when the other methods have close to zero JSD. Achieving low JSD for PR should be relatively easy, as it only has 3 categorical values, TCP, UDP, and ICMP, but the noises amplified under DP-SGD (explained in Section 3.1) significantly degrade the data fidelity after synthesized with NetShare. The normalized EMDs are all similar, due to the raw EMDs are either very large (so close to 0.9) or very small (so close to 0.10). For CAIDA, NetDPSyn is only slightly worse than NetShare for PAT, and we speculate the time-series GAN used by NetShare offsets the noises added by DP-SGD. Notice that PrivMRF is not shown in Figure 6, as it exceeds our memory limit.

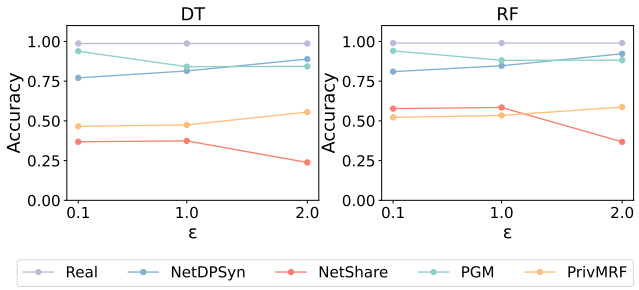

Figure 7: TON (NetFlow) accuracy with different ε value. The higher the better.

F ABLATION STUDY

Noise scale. In the default setting, we choose $\varepsilon = 2.0$ and run all the experiments. Here, we reduce ε to 1.0 and 0.1, which means the privacy protection becomes stronger, to assess its impact on the data fidelity. In Figure 7 shows the accuracy under different ε on DT and RF measurements of TON. The result shows that NetDPSyn achieves consistent accuracy with much lower ε with DT and RF.

Next, we conduct a more comprehensive comparison between NetDPSyn and NetShare, with a much larger range of ε (4.0, 8.0, 16.0, 32.0, 64.0, 10^3 and 10^{10}). We are interested in whether NetShare is able to match the performance of NetDPSyn with very large ε. In Table 6, we show the RF and DT classification accuracy on TON. For NetDPSyn, its accuracy climbs up to 0.94 after ε reaches to 16.0. For NetShare, when $\varepsilon = 10^{10}$, accuracy of DT and RF accuracy increase noticeably, but they do not exceed 0.4 and 0.6 under very large ε (10^{10}). Noticeably, NetShare without DP has around 0.6 accuracy on TON dataset (Figure 12 in [66]). In Table 7, we show the result on another dataset UGR16, but the accuracy does not change much under very large ε.

Comparison between GUMMI and GUM. For record synthesis, we propose GUMMI to improve the efficiency. The key motivation is that when the update iteration is set to 200 (the default value used by PrivSyn), GUM consumed approximately *90%* of the total time for a single experiment run.

Here, we choose the task of classification on TON as an example to compare GUMMI and GUM. We choose 7 values for update iterations, {1, 2, 3, 4, 5, 10, 20}, and run NetDPSyn with GUMMI and GUM separately. The classification accuracy corresponding to each setup is shown in Figure 8. At the initial update iterations, the accuracy observed under GUM is significantly lower than GUMMI (0.45 vs 0.85 for the decision tree). The accuracy is similar after 10 update rounds. As such, when record synthesis takes too long, especially for large network datasets, the operator can choose smaller update iteration numbers to obtain the results faster. This is especially helpful when the operator needs to search all possible parameters to find the best privacy-utility tradeoff in synthesis.

G PRIVACY ANALYSIS UNDER INFERENCE ATTACKS

We follow the basic MIA method [65] to attack the models (i.e., classifiers) trained on the raw dataset of TON, and we achieve 64.01% attack accuracy. We also tested MIA on the TON dataset synthesized by NetDPSyn, and the attack accuracy drops to 55.87% at $\varepsilon = 2$. Under $\varepsilon = 0.1$, it further drops to 40.85%. Since NetDPSyn follows record-level DP, for a synthesized packet dataset, it provides per-packet guarantee, which might not offer practical privacy guarantee. Different DP notions might be needed, e.g., user-level DP [33], and we leave the analysis as a future work.

Chanyaswad et al. [17] conducted a similar privacy analysis by computing attack accuracy under MIA on the raw and their synthesized dataset (Realistic Sensor Displacement dataset). Similar results were obtained (around 72% and 50% attack accuracy on raw and $\varepsilon = 2$ synthesized data, as shown in Figure 2 of [17]), and they conclude that data synthesis is effective to contain privacy leakage. Yet, we acknowledge that the defense can be weakened under more powerful attacks in the real-world setting [16, 50].