information

Article

Enhancing Privacy-Preserving Network Trace Synthesis Through Latent Diffusion Models

Jin-Xi Yu [1], Yi-Han Xu [1,2,*], Min Hua [1], Gang Yu [3] and Wen Zhou [4]

[1] College of Information Science and Technology & College of Artificial Intelligence, Nanjing Forestry University, Nanjing 210037, China; 8220821026@njfu.edu.cn (J.-X.Y.); min_hua@njfu.edu.cn (M.H.)
[2] School of Computer Science and Technology, Qinghai University, Xining 810000, China
[3] Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield S10 2TN, UK; gyu2@sheffield.ac.uk
[4] College of Low Altitude Equipment and Intelligent Control, Guangzhou Maritime University, Guangzhou 510725, China; wenzhou@ustc.edu
* Correspondence: xuyihan@qhu.edu.cn

Abstract

Network trace is a comprehensive record of data packets traversing a computer network, serving as a critical resource for analyzing network behavior. However, in practice, the limited availability of high-quality network traces, coupled with the presence of sensitive information such as IP addresses and MAC addresses, poses significant challenges to advancing network trace analysis. To address these issues, this paper focuses on network trace synthesis in two practical scenarios: (1) data expansion, where users create synthetic traces internally to diversify and enhance existing network trace utility; (2) data release, where synthesized network traces are shared externally. Inspired by the powerful generative capabilities of latent diffusion models (LDMs), this paper introduces NetSynDM, which leverages LDM to address the challenges of network trace synthesis in data expansion scenarios. To address the challenges in the data release scenario, we integrate differential privacy (DP) mechanisms into NetSynDM, introducing DPNetSynDM, which leverages DP Stochastic Gradient Descent (DP-SGD) to update NetSynDM, incorporating privacy-preserving noise throughout the training process. Experiments on five widely used network trace datasets show that our methods outperform prior works. NetSynDM achieves an average 166.1% better performance in fidelity compared to baselines. DPNetSynDM strikes an improved balance between privacy and fidelity, surpassing previous state-of-the-art network trace synthesis method fidelity scores of 18.4% on `UGR16` while reducing privacy risk scores by approximately 9.79%.

Keywords: network trace; diffusion models; data synthesis; privacy protection; differential privacy

Academic Editor: Jiguo Li

Received: 25 June 2025
Revised: 1 August 2025
Accepted: 7 August 2025
Published: 12 August 2025

Citation: Yu, J.-X.; Xu, Y.-H.; Hua, M.; Yu, G.; Zhou, W. Enhancing Privacy-Preserving Network Trace Synthesis Through Latent Diffusion Models. *Information* **2025**, *16*, 686. https://doi.org/10.3390/info16080686

1. Introduction

A network trace is a detailed record of data packets flowing through a computer network [1]. By capturing information about each transmitted packet, such as source and destination addresses, transmission times, and payload contents, network traces help researchers, developers, and administrators to understand the network's behavior, performance, and security properties [2]. This data can diagnose connectivity issues, identify security threats, evaluate network protocols, and improve system efficiency [3].

However, manually collecting real-world network traces can be time-consuming and labor-intensive due to their scarcity, as well as legal and ethical concerns [4]. To alleviate these issues, researchers turn to network trace synthesis to generate artificial yet representative data [5]. Synthetic traces allow for the expansion of existing datasets, helping to achieve better experimental coverage and more reliable evaluations of new algorithms, tools, and protocols. Creating larger and more diverse datasets through synthesis can reduce dependence on limited real-world traces and foster more robust and generalizable solutions.

Directly sharing raw network traces or naively generated synthetic data can reveal sensitive information, including user identities, communication patterns, or proprietary topologies [6]. Differential privacy (DP) mitigates these risks by producing synthetic datasets that retain essential statistical properties while guaranteeing formal privacy [7,8]. DP enables broader, more secure data sharing by preventing any individual's data from significantly influencing the output [9]. We focus on two scenarios for network trace synthesis.

- Data Augmentation: Users generate synthetic traces purely for internal use to diversify and expand the existing network trace for better utility.
- Data Release: The synthesized network traces are intended for external dissemination. In this scenario, directly releasing synthetic network traces risks revealing sensitive information from the original dataset. DP dataset synthesis becomes indispensable to ensure that no sensitive patterns, identities, or infrastructure details are inadvertently revealed.

By examining both scenarios, this work offers a comprehensive perspective on how network trace synthesis can effectively balance data utility with privacy requirements.

Previous network trace syntheses have often relied on heuristic-driven methods or traditional generative models (e.g., simple probabilistic distributions [10], Generative Adversarial Networks (GANs) [11,12], or Variational Autoencoders (VAEs) [13,14]). Because their representative ability is limited, models often have difficulty capturing the complexity of real-world network data. As a result, they may oversimplify patterns, fail to represent rare events, or introduce artifacts that reduce the authenticity and usability of the generated traces.

Even worse, under DP, adding noise to the training gradients of generative models [15] further reduces the practical usability of the synthetic data. Su et al. [6] proposed NetDPSyn, which first captures the underlying distributions of the original data. Then, after capturing the underlying distributions of the original data, NetDPSyn adds noise to these distributions under DP and then synthesizes network records from the resulting noisy representations.The challenge with NetDPSyn is its difficulty in managing data with large dimensions [6].

To address these limitations, we propose that NetSynDM considers using diffusion models (DMs) to generate synthetic network traces under differential privacy constraints. DMs have demonstrated stronger abilities to produce high-quality synthetic data across diverse areas than previous generative models (e.g., GANs and VAEs) on complex datasets [16]. Moreover, we still explore using latent diffusion models [17] (LDMs) in network trace synthesis. In particular, LDM can represent data in a compressed latent space, thereby reducing the dimensional size of the input data.

When integrated with DP mechanisms, NetSynDM becomes DPNetSynDM. In particular, we use DP Stochastic Gradient Descent (DP-SGD) [18] to update NetSynDM, introducing privacy-preserving noise throughout the training process. Diffusion models (DMs) operate by gradually adding and removing noise, creating complex patterns from simple distributions. Therefore, DMs naturally manage noisy intermediate representations and learn to reconstruct realistic data, they are better suited to handle privacy noise than many

other methods. As a result, even under DP constraints, DMs can still produce high-quality synthetic data, while reducing computational demands and improving the efficiency of privacy-enhancing operations.

We conduct extensive experiments to evaluate the effectiveness of NetSynDM and DPNetSynDM by comparing the fidelity and utility of their synthesized network traces against existing state-of-the-art (SOTA) methods [6]. For data augmentation, NetSynDM exhibits exceptional fidelity across multiple datasets, significantly surpassing the performance of baselines. In particular, on the `TON` dataset, it achieves a Wasserstein distance of 0.02, which is 90% lower than that of CTGAN (0.20) [19] and reduces the distance by 60% compared to GReat (0.05) [20]. This improvement highlights NetSynDM's ability to generate high-fidelity synthetic data across different datasets.

For data release to preserve privacy, although DPNetSynDM incurs a slight degradation in fidelity due to the introduction of DP, it still surpasses other DP-based baselines. For instance, in terms of Wasserstein distance between real and synthetic network traces, on `UGR16`, it achieves 0.0162, outperforming previous state-of-the-art methods such as NetDPSyn [6], which achieves 0.03, thereby offering a 40.5% improvement in data quality while preserving privacy. Besides, DPNetSynDM presents substantial improvements in privacy preservation, compared to non-DP synthesizers. For example, on the `TON` dataset, DPNetSynDM attains a privacy risk score of 0.065, achieving a 46.2% improvement over state-of-the-art non-DP methods such as GReat (0.035) [20]. To assess data utility, experimental results confirm that both NetSynDM and DPNetSynDM consistently outperform existing baselines across all evaluated datasets, irrespective of DP application.

Our contributions are three-fold:

- This paper is the first to explore how diffusion models can enhance the synthesis of network traces. We investigate their potential for addressing the challenges associated with accurately replicating complex network patterns while ensuring scalability and flexibility;
- We enable both internal data augmentation and secure external data sharing within a DP framework, addressing key challenges in balancing data utility and privacy in real-world applications;
- Extensive experiments on five widely used network trace datasets demonstrate that NetSynDM achieves state-of-the-art fidelity, while DPNetSynDM effectively mitigates privacy risks while maintaining higher data quality than existing DP-based synthesis methods.

2. Background

We first introduce the network dataset, then discuss diffusion models (DMs) and latent diffusion models (LDMs), and finally define differential privacy (DP) in the network dataset and introduce the DP-SGD algorithm.

2.1. Network Dataset

Consistent with prior studies [6,15], our method targets the synthesis of data using header fields from network packets and flows. In many research scenarios, releasing header fields alone is often sufficient to reduce privacy risks. This approach mitigates concerns associated with exposing sensitive information, as the payload of a packet may contain personal data or other private content. We use six public datasets that contain either packet or flow data. We describe the common fields in these datasets as follows:

- Packet Header: This contains information for each observed packet across both the network (Layer 3) and transport (Layer 4) layers. Included fields typically comprise source/destination IPs (`srcip`, `dstip`), source/destination ports (`srcport`,

`dstport`), the transport protocol (`proto`, such as TCP, UDP, and ICMP), the capture timestamp (`ts`), and the packet size (`pkt_len`), along with additional fields like checksum (`chksum`) and a dataset label (`label`).

- Flow Header: A network flow is identified by a five-tuple approach ⟨`srcip`, `dstip`, `srcport`, `dstport`, `proto`⟩ [21]. Its associated header includes the timestamp of the initial packet (`ts`), the total flow duration (`td`), the packet count (`pkt`), the cumulative byte size (`byt`), and the assigned label (`label`).

2.2. Privacy Leakage from Network Dataset

Network traces can include various sensitive data, such as IP addresses, MAC addresses, and other unique identifiers that can reveal user behavior or location information [22,23]. Although releasing only the header fields (without the payload) lowers privacy risks, it does not fully eliminate them. IP addresses, for example, can still reveal private details. To address this, data anonymization and data synthesis remain the main approaches. Anonymization techniques, such as CryptoPan's prefix-preserving anonymization [24], are widely applied to IP addresses. However, a recent study showed that anonymization can still be vulnerable to de-anonymization attacks if the institution behind an IP prefix engages in sensitive or controversial activities (e.g., sending emails to a sensitive organization) [25]. In contrast, data synthesis offers more fine-grained control over balancing privacy and utility [15].

By generating synthetic network traces, we can achieve stronger privacy safeguards while keeping the data useful for further analysis. Our goal is to establish provable privacy guarantees for these synthesized traces, thereby enhancing privacy protection without affecting the quality of subsequent research or analysis.

2.3. Diffusion Models

2.3.1. Standard Diffusion Models

Diffusion models (DMs) [26,27] belong to a family of likelihood-based generators that utilize both forward and backward Markov transitions. During the diffusion (forward) phase, random noise is gradually injected into an initial sample x_0, which is drawn from $q(x_0)$:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}), \tag{1}$$

The added noise comes from fixed distributions $q(x_t|x_{t-1})$ with pre-specified variances $\{\beta_1, \ldots, \beta_T\}$. At each time step t, standard Gaussian noise $z \sim \mathcal{N}(0, I)$ is introduced, progressively degrading the original sample. Conversely, the backward process gradually recovers x_0 from the noisy version $x_T \sim q(x_T)$, as modeled by

$$p(x_{0:T}) = \prod_{t=1}^{T} p(x_{t-1}|x_t). \tag{2}$$

As the true backward transition $p(x_{t-1}|x_t)$ is analytically intractable, a neural network parameterized by θ is used to approximate it by optimizing a variational lower bound:

$$\begin{aligned} \log q(x_0) \geq \mathbb{E}_{q(x_0)}[\log p_\theta(x_0|x_1)] - \mathrm{KL}(q(x_T|x_0)\|q(x_T)) \\ - \sum_{t=2}^{T} \mathrm{KL}(q(x_{t-1}|x_t, x_0)\|p_\theta(x_{t-1}|x_t)). \end{aligned} \tag{3}$$

Unlike GANs or VAEs, diffusion models do not generate samples directly from noise vectors. Instead, DMs recover data in multiple steps through iterative denoising. Each backward step corrects the sample incrementally, enhancing model convergence and reducing overfitting. Depending on the loss type, the model may estimate the clean data

from noisy input (denoising) or directly recover the noise (noise-prediction) [28]. The latter yields the common noise prediction loss:

$$\mathcal{L}_{\text{sdm}}(\theta) = \mathbb{E}_{(x_t,\epsilon),t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2\right], \tag{4}$$

where x_t is the noisy data at time step t, ϵ is the injected noise, and ϵ_θ is the neural network that predicts the noise in the original data space. While diffusion models have achieved remarkable success in various data modalities [29–31], their application in network traffic synthesis privacy preservation remains limited [1].

While diffusion models have achieved remarkable success in various data modalities [29–31], their application in network traffic synthesis and privacy preservation remains limited. Recent studies such as NetDiffusion [32], NetDiff [33], HotNets [1], and Chai et al. [34] have explored diffusion-based methods for generating network traffic. However, these approaches primarily aim to achieve high-fidelity synthesis, protocol consistency, or hierarchical modeling for service-aware traffic, without addressing data privacy concerns. Specifically, NetDiffusion introduces protocol-constrained traffic augmentation, NetDiff proposes a hierarchical diffusion architecture for service-guided flow generation, HotNets focuses on unconditional fidelity, and Chai et al. [34] generates mobile traffic for network planning based on open datasets. In contrast, our work uniquely targets the problem of privacy-preserving network trace synthesis. By integrating LDMs with DP mechanisms, our method not only captures complex dependencies in high-dimensional network data but also offers provable privacy guarantees. To the best of our knowledge, our approach is the first to bridge LDMs with DP for network trace synthesis, enabling a fine-grained balance across fidelity, utility, and privacy objectives.

2.3.2. Latent Diffusion Models

LDMs [35] are an improvement upon standard DMs [27]. LDMs utilize an autoencoder architecture [35], consisting of an encoder Enc and a decoder Dec, to transform high-dimensional data x into low-dimensional latent representations z, and they then use the decoder to reconstruct the original data. This autoencoder [35] combines perceptual loss and adversarial objectives with additional regularization to better control the variance in the latent space.

LDMs improve training efficiency by performing the diffusion process within the latent representation space of pretrained autoencoders, which greatly lowers the computational overhead compared to diffusion in raw pixel space. Furthermore, with the integration of attention mechanisms, LDMs enable versatile conditional generation based on inputs like text descriptions or category labels [35]. During training, a neural network parameterized by τ_θ predicts initial noise from the noisy latent representations. The parameters $\theta = [\theta_U, \theta_{\text{Attn}}, \theta_{\text{Cn}}]$ are optimized by minimizing the prediction error defined by the following formula:

$$\mathcal{L}_{\text{ldm}}(\theta) = \mathbb{E}_{(z_t,y),\tau,t}\left[\|\tau - \tau_\theta(z_t, t, y)\|_2^2\right] \tag{5}$$

Once training is completed, the latent representations $\tilde{z}$ are mapped back to the original image domain using the decoder.

2.4. Differential Privacy

Differential privacy [36] provides a rigorous mathematical foundation for quantifying privacy within statistical databases. It ensures robust privacy protection by limiting how much the inclusion or exclusion of one record can influence the result of a statistical query. In practice, this is implemented by injecting calibrated statistical noise into the computation.

Definition 1 ((ϵ, δ)-Differential Privacy [37]). *For $\epsilon, \delta > 0$, an algorithm M is (ϵ, δ)-differentially private if, for any pair of neighboring databases X', X'' and any subset S of possible outputs produced by M,*

$$\Pr[M(X') \in S] \leq e^{\epsilon} \Pr[M(X'') \in S] + \delta. \tag{6}$$

In this context, e^{ϵ} acts as a multiplicative bound on the ratio of probabilities, controlling their divergence. Smaller ϵ values imply tighter bounds, limiting how much probabilities differ between neighboring datasets. Therefore, ϵ quantifies privacy, where a lower value implies stronger protection. Typically, δ is set to 10^{-5}, allowing for occasional failures in privacy guarantees to ensure practical feasibility [36].

DP-SGD. Gradient descent is a fundamental technique for training models under differential privacy constraints. The goal is to reduce the loss function L, which measures how much the predictions deviate from the ground truth. The parameters θ are updated iteratively using the gradient $\nabla_\theta L$, scaled by the learning rate η:

$$\theta \leftarrow \theta - \eta \cdot \nabla_\theta L. \tag{7}$$

To enable DP-compliant learning, Song et al. [38] developed the DP-SGD algorithm, which modifies standard SGD. At each iteration, the gradients are first computed and then modified in two ways before updating the model: the L_2 norm is clipped to a predefined threshold $C \geq 1$, and Gaussian noise sampled from a multivariate distribution is added, scaled by both the noise multiplier σ and the clipping bound C:

$$\theta \leftarrow \theta - \eta \cdot \text{CLIP}(\nabla_\theta L, C) + \mathcal{N}\left(0, \frac{C^2\sigma^2}{I}\right) \tag{8}$$

Gradient clipping restricts the influence of individual samples by capping their gradient norms, and the added noise helps to obscure specific information from being leaked about any single data point. These mechanisms provide DP guarantees for DP-SGD. Gradually, clipping, noise addition, and updates are typically applied at the end of each batch to maintain performance. Superior optimizers like the Adam optimizer can be used with DP-SGD by simply substituting the gradient updates while still preserving privacy guarantees [39]. Modern implementations also use Laplacian noise and Rényi DP for better performance and tighter privacy bounds [40–42].

3. Real-World Scenarios

This section highlights two key real-world application scenarios of network synthesis, emphasizing the significance of advancing this area in our paper:

- Data Expansion: Data expansion addresses issues such as data scarcity and certain data's rarity during training. When the training dataset is insufficient, the model's generalization ability is severely impacted [43], leading to overfitting or prediction bias. Data expansion solves these issues by increasing the size, coverage, or diversity of the dataset, which is especially important in scenarios where data collection is difficult or expensive. In the network traffic domain, generative approaches can be used for data expansion, producing synthetic traces that improve generalization, support rare-event modeling, and enhance the robustness of downstream predictive models [1].
- Data Release: As privacy regulations become more stringent, data release has become particularly important, ensuring that data sharing does not violate privacy protection laws. Releasing raw data can lead to personal privacy leaks or legal violations [44]. Therefore, using DP techniques or synthetic data release allows for data sharing and

analysis without exposing sensitive information and permits data release for research, collaboration, or commercial analysis [45]. This is crucial for complying with privacy regulations, protecting user privacy, and promoting data sharing. Data release helps to resolve the conflict between data sharing and privacy protection, promoting research and collaboration.

4. Methodology

4.1. Motivation and Workflow

In the context of data sharing and privacy protection, conventional generative models like GANs and VAEs face significant privacy risks when synthesizing network data. Although these models can generate plausible data samples, they are vulnerable to linkage attacks, where adversaries may infer sensitive information from synthetic data by identifying correlations with original records. Researchers have attempted to integrate DP mechanisms with generative models to strengthen privacy protection and prevent personal information leakage. To address this, researchers have incorporated DP mechanisms into generative frameworks, typically through gradient clipping and noise injection (e.g., DP-SGD). However, these methods often degrade model performance, introducing excessive noise that compromises fidelity and utility. For instance, while DP-SGD ensures strong privacy, it notably increases the Earth Mover's Distance (EMD)—a widely used fidelity metric—from 0.10 (no DP-SGD) to 0.35 (with DP-SGD), even when the privacy budget is relatively high ($\epsilon = 24.24$), as shown in Table 5 of [15]. While DP mechanisms enhance privacy protection, the excessive noise introduced reduces the usability and authenticity of the synthetic dataset.

To mitigate this challenge, we proposed integrating DP with diffusion models (DMs) to generate network data. Diffusion models offer a unique denoising-based framework that progressively adds and removes noise across multiple timesteps. This iterative generation process allows DMs to capture intricate temporal and structural patterns in network traces, including rare events or bursty behaviors that simpler models may overlook. Moreover, compared to conventional models, diffusion models demonstrate superior scalability in high-dimensional settings. By decoupling the generation into step-wise transformations with shared parameters, DMs reduce the optimization burden and better preserve correlations across diverse features. This property is particularly valuable for modeling network traces, which often consist of heterogeneous features such as protocol types, port distributions, and inter-arrival times. Overall, the diffusion-based framework enables a more flexible balance between fidelity, utility, and formal privacy guarantees, making it a compelling choice for secure and realistic network trace synthesis.

Compared with NetDPSyn, which primarily relies on histogram-based distribution modeling and marginal sampling, our method leverages a continuous-time denoising framework to better capture temporal dependencies and rare communication patterns. This distinction enables improved fidelity and scalability, especially in high-dimensional network trace settings.

4.2. Preprocessing

Building on the ideas of NetDPSyn [6], we implemented a more fine-grained strategy. Specifically, network fields were grouped independently according to their statistical distributions and attribute types. As a first step, we applied customized binning rules depending on the attribute type, covering five major field categories:

1. IP (`srcip`, `dstip`): IPs with low frequency are aggregated using the /30 prefix.
2. Port (`srcport`, `dstport`): Ports below 1024 are reserved as exceptions, while others are grouped in steps of 10.

3. Categorical attributes (e.g., `proto`, `label`): These are left unchanged due to their limited domain sizes.
4. Numerical attributes (`pkt`, `byt`, `td`): A logarithmic scale is applied to discretize these values,
$$\log(1+x), \tag{9}$$
which effectively reduces the number of bins compared to linear segmentation.
5. Timestamp (`ts`): A separate strategy is adopted for timestamps, as discussed later in Section 4.3.

After type-based partitioning, we applied frequency-aware grouping to merge bins with low occurrence. While frequency information was derived from the raw dataset, our approach only used global frequency counts as aggregated statistics without accessing individual user-level records. Therefore, this procedure did not violate differential privacy principles. Importantly, no privacy budget was consumed during preprocessing, since all operations were deterministic or non-interactive and did not involve any randomized mechanism over sensitive data.

Moreover, the preprocessing stage only transformed the raw input into a discretized and structured representation, without involving any model parameters or learning process. The generative model (diffusion network) did not access the raw data directly, and privacy protections were entirely enforced during training using DP-SGD. This design was consistent with standard DP-compliant machine learning pipelines, as static and non-adaptive preparation steps did not incur formal privacy cost. The entire privacy budget was instead allocated to the training stage, as discussed in Section 4.6.

4.3. NetSynDM: Highly Efficient Network Trace Synthesis

To synthesize high-fidelity network trace data while preserving privacy and addressing the heterogeneity of network data, we proposed NetSynDM, a novel latent diffusion model (LDM) framework tailored for network data synthesis. In contrast to standard diffusion models that operate directly in the input space, our method first mapped discrete, binarized network attributes into a compact latent space through embedding layers and a shallow encoder. The denoising process was then applied in this latent space, improving both the modeling efficiency and synthesis quality.

Diffusion models are renowned for their ability to model complex high-dimensional data distributions, offering significant advantages over traditional generative methods (e.g., GANs) in capturing intricate dependencies among network attributes. Network data, which comprises categorical, numerical, and temporal attributes, requires specialized preprocessing to fit into a diffusion-based synthesis pipeline.

The overall workflow of NetSynDM is summarized in Algorithm 1. We followed the two-stage preprocessing pipeline proposed by NetDPSyn [6], consisting of type-dependent binning and frequency-aware bin merging for sparsity reduction. Although optional noise could be added to frequency counts, this step did not consume any privacy budget, as explained in Section 4.2. In the type-dependent binning stage, attributes were discretized based on domain-specific characteristics. For example, low-frequency IP addresses were aggregated by the /30 prefix, port numbers below 1024 retained their individual values, and higher ports were binned at intervals of 10, as shown below:

$$\text{bin}(p) = \lfloor (p - 1024)/10 \rfloor + 1024, \quad \text{for } p \geq 1024 \tag{10}$$

Categorical attributes were directly encoded as integers, while numerical attributes were binned using logarithmic transformation to handle skewed distributions. Here, Δ is a resolution parameter that controls bin granularity in the log space:

$$\text{bin}(x) = \lfloor \frac{\log(1+x)}{\Delta} \rfloor \tag{11}$$

Timestamps were binned relative to a reference point at fixed intervals. We introduced τ as the interval width for timestamp discretization:

$$\text{bin}(ts) = \lfloor (ts - ts_0)/\tau \rfloor \tag{12}$$

This step converted the raw data into a discrete representation:

$$z = [z_1, z_2, \ldots, z_d] \tag{13}$$

where z_i denotes the bin ID for the i-th attribute.

To further reduce sparsity and protect privacy, we introduced a DP mechanism that merged low-frequency bins. Gaussian noise was added to each attribute's bin frequency $f(b)$ to satisfy (ϵ, δ)-DP:

$$\tilde{f}(b) = f(b) + \text{N}(0, \sigma^2), \quad \sigma = \frac{\Delta f \sqrt{2\ln(1.25/\delta)}}{\epsilon} \tag{14}$$

The perturbed low-frequency bins were merged to generate a refined discrete representation z'. Since diffusion models operate in continuous space, we converted z' into latent representations by mapping each bin ID z'_i to a k-dimensional vector e_i through an embedding layer:

$$e_i = \text{Emb}(z'_i), \quad e_i \in \mathbb{R}^k \tag{15}$$

These vectors were concatenated to form the initial data point:

$$x_0 = [e_1, e_2, \ldots, e_d] \in \mathbb{R}^{d \times k} \tag{16}$$

where d is the number of attributes and $k = 64$. We employed a Gaussian diffusion process with $T = 1000$ steps to synthesize network traces. The forward process gradually added noise to x_0, modeled as

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; (1-\beta_t)x_{t-1}, \beta_t I) \tag{17}$$

The reverse process was parameterized by a neural network $p_\theta(x_{t-1} \mid x_t)$, formulated as

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{18}$$

The model was trained by predicting the noise ϵ, ensuring effective denoising and preserving statistical properties, with the following objective:

$$L = \mathbb{E}_{x_0, \epsilon, t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2 \right] \tag{19}$$

The denoising network ϵ_θ adopted a Transformer architecture, taking x_t with enhanced timestep t information as the input to capture long-range dependencies across attributes. To generate synthetic traces, we sampled from the noise distribution using

$$x_T \sim \mathcal{N}(0, I) \tag{20}$$

and generate x'_0 via the reverse process. The embedded vectors e'_i were then mapped back to bin IDs z'_i as follows:

$$z'_i = \arg\min_z \|e'_i - \text{Emb}(z)\|_2 \tag{21}$$

The final transformation restored the original attribute values and enforced domain constraints, ensuring that the synthesized data matched the real data format with high fidelity. This approach integrated preprocessing, diffusion processes, and data generation into a complete pipeline suitable for network data synthesis scenarios requiring high fidelity and privacy protection.

Algorithm 1 The Workflow of NetSynDM

Require: Network trace data sample

$x = [x_{ip}, x_{port}, x_{cat1}, \ldots, x_{catC}, x_{num}, x_{ts}]$

Ensure: Synthesized network trace x_{out}

Type-Dependent Binning:

For $p \geq 1024$: $\text{bin}(p) = \lfloor (p - 1024)/10 \rfloor + 1024$

For numerical x: $\text{bin}(x) = \lfloor \log(1 + x)/\Delta \rfloor$

For timestamp ts: $\text{bin}(ts) = \lfloor (ts - ts_0)/\tau \rfloor$

Categorical: directly encode as integers

Form discrete vector: $z = [z_1, z_2, \ldots, z_d]$

Frequency-Aware Bin Merging (no DP budget consumed):

Operates on global frequency statistics; optionally perturbed but privacy cost is negligible.

Add Gaussian noise:

$$\tilde{f}(b) = f(b) + \mathcal{N}(0, \sigma^2), \quad \sigma = \frac{\Delta f \sqrt{2 \ln(1.25/\delta)}}{\epsilon}$$

Merge low-frequency bins to form refined vector z'

Embedding:

$$e_i = \text{Emb}(z'_i), \quad e_i \in \mathbb{R}^k$$

$x_0 = [e_1, e_2, \ldots, e_d] \in \mathbb{R}^{d \times k}$

Forward Diffusion: For $t = 1$ to T:

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; (1 - \beta_t)x_{t-1}, \beta_t I)$$

Reverse Diffusion:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

Training Objective:

$$L = \mathbb{E}_{x_0, \epsilon, t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2\right]$$

Sampling and Generation:

$$x_T \sim \mathcal{N}(0, I)$$

Run reverse steps to obtain x'_0

Decode:

$$z'_i = \arg\min_z \|e'_i - \text{Emb}(z)\|_2$$

Restore attributes from z' using domain rules

return Synthesized data x_{out}

4.4. DPNetSynDM: Privacy-Preserving for NetSynDM

The complete procedural steps of DPNetSynDM are presented in Algorithm 2. DP-NetSynDM extends the NetSynDM framework by embedding explicit differential privacy mechanisms throughout the diffusion and synthesis phases. While NetSynDM applies

limited DP logic during preprocessing, DPNetSynDM allocates the entire privacy budget to the training phase and further enhances privacy robustness by additionally employing the following:

- DP during Training: Gradients are clipped and perturbed with calibrated Gaussian noise:
$$\tilde{g}(\theta) = g(\theta) + \mathcal{N}(0, C^2\sigma_g^2) \tag{22}$$
where CC denotes the clipping norm.
- DP during Data Synthesis: Embeddings receive additional noise during the reverse diffusion process as follows:
$$\tilde{e}'_i = e'_i + \mathcal{N}(0, \sigma_e^2 I) \tag{23}$$

This comprehensive DP strategy significantly strengthened privacy guarantees without leading to considerable degradation in the synthesized data quality, achieving superior balance compared to existing DP-integrated generative models.

Algorithm 2 The Workflow of DPNetSynDM

Require: Network trace data sample
$x = [x_{ip}, x_{port}, x_{cat1}, \ldots, x_{catC}, x_{num}, x_{ts}]$
Ensure: Differentially private synthesized data x_{out}^{DP}
Step 1: Inherit Preprocessing from NetSynDM:
Apply type-dependent and frequency-dependent binning using global frequency statistics.
No privacy budget is consumed, as this step uses only aggregated data and deterministic operations.
Step 2: Forward Diffusion: Add Gaussian noise over T steps:
$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; (1-\beta_t)x_{t-1}, \beta_t I)$$
Step 3: DP during Training: Apply gradient clipping and calibrated Gaussian noise:
$$\tilde{g}(\theta) = g(\theta) + \mathcal{N}(0, C^2\sigma_g^2)$$
where C denotes the clipping norm
Step 4: Reverse Diffusion: Use Transformer to predict noise and reverse the process:
$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$
Step 5: DP during Data Synthesis: Inject additional noise into the output embeddings:
$$\tilde{e}'_i = e'_i + \mathcal{N}(0, \sigma_e^2 I)$$
Step 6: Decode: Find the closest bin ID:
$$z'_i = \arg\min_z \|\tilde{e}'_i - \text{Emb}(z)\|_2$$
Step 7: Restore Attributes: Convert bin IDs z' back to raw values with domain-specific rules.
return Differentially private synthesized data x_{out}^{DP}

4.5. Model Tuning and Training

An essential step in our synthesis framework involved determining effective hyperparameters for the data generators. Although some synthesis methods overlook this phase, prior research [46] indicates that appropriate tuning strategies can significantly enhance generator performance.

To streamline hyperparameter selection, we incorporated a dedicated tuning phase into our evaluation workflow. This phase was guided by a unified objective function, L, which balanced multiple performance dimensions:

$$L(A) = \alpha_1 \text{Fidelity}(A) + \alpha_2 \text{MLA}(A) + \alpha_3 \text{QueryError}(A) \tag{24}$$

The weights α_1, α_2, and α_3 corresponded to fidelity (measured via Wasserstein distance), machine learning adaptability, and query accuracy, respectively. For consistency across experiments, all weights were set to 1. This strategy was found to improve the overall realism of the synthesized data compared to untuned baselines. To solve the tuning problem, we utilized Optuna [47] to minimize L.

In practice, Optuna was applied to PrivSyn, LDM, NetSynDM, and DPNetSynDM, ensuring that each model was optimized under consistent conditions for the benchmark mixed-type dataset. Each synthesizer underwent 30 optimization trials to identify the best-performing hyperparameter set.

After hyperparameters were finalized, model training proceeded accordingly. Different synthesis models leverage varied network structures and optimization targets at this stage.

4.6. Privacy Budget Allocation

We incorporated differential privacy (DP) into our data synthesis pipeline following the DP-SGD framework and explicitly clarified how the privacy budget was allocated across the following different stages:

- Preprocessing stage: This included tokenization, binning, and embedding initialization. Although frequency-based binning was applied, it only used global frequency counts derived from the raw dataset and did not access individual user-level records. All operations were deterministic and non-interactive, meaning that they did not adapt based on input data or model feedback. Therefore, no privacy budget was consumed at this stage. Furthermore, the generative model did not access raw data directly, and all DP guarantees were enforced during training. This design was aligned with standard DP-compliant pipelines, where static data preparation steps are not counted toward privacy cost.
- Model training stage: The entire privacy budget was consumed during training. We employed the DP-SGD optimizer with fixed gradient clipping and calibrated Gaussian noise injection. The cumulative privacy loss ε was computed using the Rényi Differential Privacy (RDP) accountant [42].
- Data synthesis stage: No additional privacy cost was incurred during data generation, as this step relied solely on the trained model. According to the post-processing invariance property of DP, this stage did not contribute to further privacy leakage.

In all experiments, both NetDPSyn and DPNetSynDM were trained under the same privacy constraint of $\varepsilon = 1.0$, enabling a fair comparison of different synthesis strategies under equivalent privacy guarantees.

5. Experimental Settings

5.1. Datasets

To ensure reproducibility, we selected six widely used public datasets, comprising three flow header and three packet header traces. These datasets varied in terms of deployment environments, data collection mechanisms, and time spans.

For flow-based datasets, we extracted 11 key attributes from each record, including the (1) source IP, (2) destination IP, (3) source port, (4) destination port, (5) protocol type,

(6) flow start time, (7) duration, (8) number of packets, (9) number of bytes, (10) label indicating benign or malicious activity, and (11) specific attack type (e.g., DoS, brute force, scanning). For packet-level datasets, we focused on the IP header along with arrival timestamp and transport-layer port numbers (TCP/UDP only).

The data was initially split as follows: 80% for training and 20% for testing. Subsequently, 20% of the training portion was reserved as a validation set to facilitate hyperparameter optimization. Table 1 summarizes the key characteristics of each dataset.

Table 1. An overview of the datasets utilized in this study. The domain size is calculated as the total across all attribute domains.

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

5.2. Baselines

This paper introduces two types of network synthesis, NetSynDM and DPNetSynDM, which operate without and with DP, respectively. Thus, for NetSynDM, we compared it with traditional network synthesis methods, i.e., CTGAN [19], TVAE [19], and GReat [20], which are widely studied marginal-based synthesis approaches used in tabular synthesis. We adapted them for network trace synthesis. In addition, we considered Private-PGM [48] and PrivMRF [49], two classical marginal-based DP synthesis methods. However, these models are no longer considered state-of-the-art methods under current benchmarks, as recent studies have revealed limitations in capturing complex dependencies in high-dimensional or structured datasets such as network traces. We did not include them in our experimental comparisons:

- CTGAN is a no-DP synthesis algorithm that can be applied to network traffic data. It leverages GANs to learn the distributions of network traffic data. Throughout the training process, methods like conditional generation and the use of Wasserstein loss [50] are adopted to improve the realism and variety of generated traffic data.
- TVAE addresses the challenges of non-Gaussian continuous distributions in network traffic data by applying mode-specific normalization, thereby generating more accurate and realistic synthetic traffic data.
- GReat is a data synthesis method based on an LLM that can be adapted or extended to network traffic data by converting the various fields into LLM-friendly textual representations before generating synthetic data.

Then, for DP network synthesis, we selected two state-of-the-art methods, NetShare [15] and NetDPSyn [6], which are defined as follows:

- NetShare uses GANs to learn generative models to generate synthetic packets automatically under DP;
- NetDPSyn is a non-parametric DP synthesizer that iteratively updates the synthetic dataset to align with the target noise marginals.

5.3. Implementations

All algorithms presented in this paper, including the baseline models and the key algorithms NetDPSyn, NetSynDM, and DPNetSynDM, were implemented using Python 3.9. The performance of these synthesizers was evaluated based on the following dimensions:

Fidelity. We adopted the Wasserstein distance to quantify fidelity by assessing the distributional gap between real and synthetic data. Depending on the analysis context, the real dataset D may refer to either D_{train} or D_{test}. In our experiments, we utilized the POT library [51] to calculate all one-way and two-way marginal distributions, with the average serving as the final fidelity metric.

Privacy. We used the membership disclosure score (MDS) [52] as a privacy metric to estimate the membership disclosure risk of data synthesizers. However, directly computing the MDS on the full dataset is computationally prohibitive, as it would require retraining a separate model for each individual record. To mitigate this issue, we adopted a practical approximation strategy inspired by the method proposed in [53]. Specifically, we let $\mathcal{D}_{\text{sub}} \subset D_{\text{train}}$ denote a randomly selected subset of the training data. We trained m models, with each excluding a unique holdout instance from $\mathcal{D}_{\text{sub}}$. The MDS was then approximated by averaging the prediction advantage observed across these m holdout cases. While this strategy substantially reduced the computational cost, it introduced two potential limitations: (1) the estimated risk may have exhibited higher variance due to the limited sample size, and (2) the estimate may have been conservative, i.e., it may have underestimated the true MDS, if the held-out instances were not representative of highly memorized or outlier records. Nevertheless, this approximation provided a tractable and empirically meaningful upper bound on the disclosure risk and facilitated fair comparisons across different synthesis methods under consistent evaluation conditions.

Utility. To assess the practical effectiveness of the synthetic data, we adopted machine learning affinity (MLA) and query error as core utility metrics. MLA was computed using a diverse set of seven machine learning models: Logistic Regression (or Ridge Regression), Decision Trees, Random Forests, Multi-Layer Perceptrons (MLP), XGBoost, CatBoost [54], and Transformers [55]. All models were carefully fine-tuned on the original training set to achieve optimal performance. For classification and regression evaluation, we used the F1 score and Root Mean Squared Error (RMSE) as the evaluation criteria. To evaluate the query error, we constructed 1000 random three-attribute query conditions and executed range or point queries on both real and synthetic datasets.

5.4. Experimental Environment

All experiments were conducted on a cluster of servers equipped with Ubuntu 22.04.5 LTS (Linux 5.15.0-107-generic). The cluster consisted of three physical servers, equipped, respectively, with four NVIDIA A6000 GPUs (NVIDIA Corporation, Santa Clara, CA, USA), two NVIDIA RTX 3090 GPUs (NVIDIA Corporation, Santa Clara, CA, USA), and four NVIDIA RTX 4080 GPUs (NVIDIA Corporation, Santa Clara, CA, USA), ensuring the stability and reproducibility of the results.

5.5. Computational Cost Considerations

While evaluating the performances of data synthesizers, it is also important to consider their computational cost. We compared the training times of representative methods on the DC dataset (1 M records, 40.6 MB) using the same hardware (NVIDIA RTX 4080 16 GB). As shown in Table 2, traditional non-DP synthesizers like CTGAN, TVAE, and GReaT took 3–9 h to train, while diffusion-based methods such as NetSynDM and DPNetSynDM had faster training times of 1.5–3.5 h due to their stable and parallelizable sampling processes. However, the addition of differential privacy mechanisms (e.g., noise injection and adaptive binning) in DPNetSynDM increased its training time significantly compared to that of NetDPSyn, which is based on histogram modeling and does not rely on deep generative training.

This trade-off highlights that while diffusion models, especially those with DP, are computationally heavier, their utility in high-stakes privacy-sensitive applications often justifies the additional cost.

Table 2. Training time comparison on the DC dataset (1M records, 40.6 MB) using identical hardware (NVIDIA RTX 4080 16 GB). Among non-DP methods, Transformer-based GReaT incurs the highest training time. NetSynDM achieves efficiency comparable to the LDM while preserving fidelity. DP-based models introduce additional overhead due to privacy-preserving mechanisms, with DP-NetSynDM incurring the highest cost among them due to iterative denoising under DP constraints.

Method	Time (hh:mm)	Privacy
CTGAN	03:06	No
TVAE	05:00	No
GReaT	09:36	No
LDM	01:30	No
NetSynDM	01:31	No
NetDPSyn	00:30	Yes ($\varepsilon = 1.0$)
DPNetSynDM	03:30	Yes ($\varepsilon = 1.0$)

6. Results Analysis

6.1. Fidelity Evaluation

We evaluate the fidelity of data synthesis by computing the Wasserstein distance on the training dataset D_{train} (shown in Table 3). The results indicate that among no-DP synthesizers, TAVE and NetSynDM achieve the lowest Wasserstein distances across most datasets, demonstrating superior fidelity. In contrast, deep generative models such as CTGAN and DPNetSynDM show significantly higher distances, suggesting a notable fidelity gap compared to other no-DP synthesizers.

Table 3. Fidelity evaluation (i.e., Wasserstein distance, where a lower score indicates higher quality) of synthesis algorithms on D_{train}. The privacy budget ϵ of no-DP synthesizers is set to ∞, and it is set to 1 for DP methods.

Methods	TON	DC	CAIDA	CIDDS	UGR16
CTGAN	0.1994 ± 0.0042	0.0414 ± 0.0055	0.0894 ± 0.0034	0.3292 ± 0.0088	0.2652 ± 0.0041
TAVE	0.0391 ± 0.0041	0.0474 ± 0.0033	0.0188 ± 0.0028	0.0202 ± 0.0028	0.0341 ± 0.0044
GReat	0.0490 ± 0.0079	0.0210 ± 0.0030	0.0412 ± 0.0024	0.0316 ± 0.0032	0.0409 ± 0.0042
LDM	0.0509 ± 0.0055	0.0112 ± 0.0023	0.0185 ± 0.0039	0.0202 ± 0.0049	0.0255 ± 0.0064
NetSynDM	0.0208 ± 0.0035	0.0102 ± 0.0017	0.0423 ± 0.0047	0.0156 ± 0.0038	0.0122 ± 0.0017
NetDPSyn	0.0294 ± 0.0032	0.1165 ± 0.0080	0.1481 ± 0.0096	0.0650 ± 0.0052	0.0272 ± 0.0036
DPNetSynDM	0.1304 ± 0.0066	0.0911 ± 0.0036	0.0807 ± 0.0028	0.1168 ± 0.0056	0.0162 ± 0.0034

Among differential privacy (DP) synthesizers (shown in Figure 1), NetDPSyn achieves the best fidelity performance, maintaining relatively low Wasserstein distances across most datasets. However, DP-based models suffer a substantial fidelity drop compared to their no-DP counterparts, highlighting the inherent trade-off between privacy and data quality. Notably, DPNetSynDM shows the highest fidelity loss, particularly in the CIDDS datasets, where Wasserstein distances are considerably higher than for other methods.

Examining the fidelity of different attribute types reveals that statistical models, such as LDM, achieve more stable performance, particularly for categorical attributes. However, they struggle with numerical attributes, which require more complex modeling capabilities. Deep generative models like GReat and TAVE appear to better handle numerical attributes, but they do so at the cost of increased variance. This trade-off suggests that

while statistical methods maintain stability, deep generative models offer improved fidelity in high-dimensional numerical spaces. Overall, the results reinforce the known challenges in balancing privacy and fidelity. While DP-based methods like NetDPSyn mitigate privacy risks effectively, they come at the cost of significant fidelity degradation. No-DP-based methods such as TAVE and NetSynDM provide stronger fidelity performance but lack privacy guarantees. Future work should explore hybrid approaches that can bridge this gap, optimizing fidelity while adhering to strict privacy constraints.

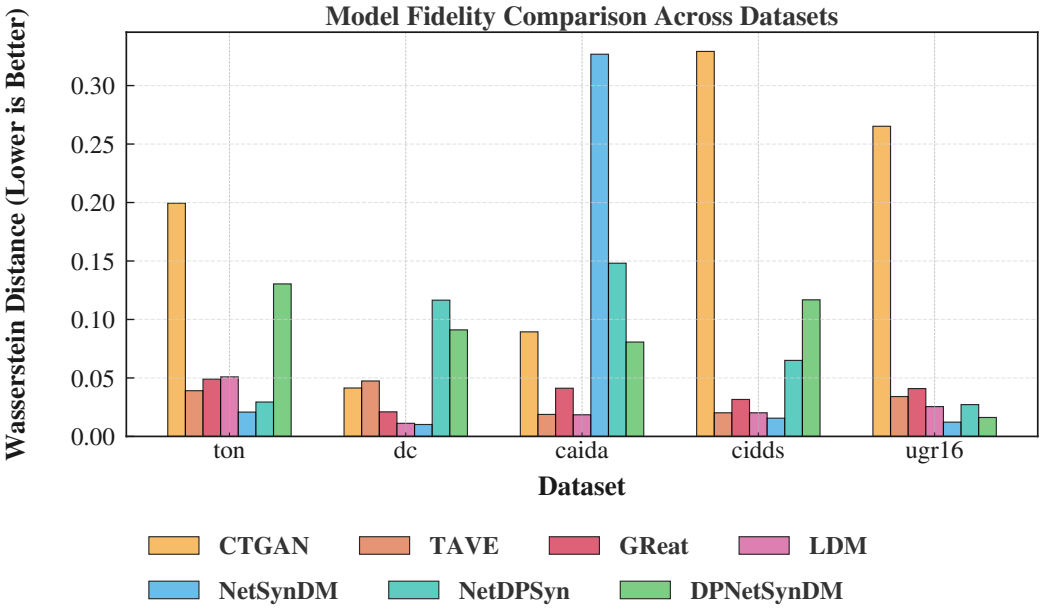

Figure 1. Model fidelity comparison across datasets.

6.2. Privacy Evaluation

Table 4 presents the privacy evaluation results of different data synthesizers. The results reveal a significant difference in privacy protection between no-DP and DP synthesizers. Unlike the fidelity evaluation results, where deep generative models such as TAVE and NetSynDM excel in fidelity, the privacy evaluation reveals a different trend. While NetDPSyn demonstrates moderate fidelity performance, it offers the strongest privacy protection across all datasets, exhibiting the lowest membership disclosure risks.

Table 4. Privacy evaluation is conducted using the MDS, where a lower value indicates better empirical privacy protection. The evaluation focuses on no-DP synthesizers with a privacy budget of $\epsilon = 1.0$. The SELF method is used as the baseline, serving as the empirical lower bound of MDS (the theoretical upper bound of MDS is 0 by definition).

Methods	TON	DC	CAIDA	CIDDS	UGR16
CTGAN	0.0264	0.0128	0.0136	0.0159	0.0164
TAVE	0.1211	0.0133	0.0145	0.0143	0.0155
GReat	0.0346	0.0168	0.0132	0.0212	0.0148
LDM	0.0208	0.0132	0.0122	0.0141	0.0146
NetSynDM	0.0196	0.0104	0.0118	0.0138	0.0142
NetDPSyn	0.0768	0.0208	0.0226	0.0327	0.0228
DPNetSynDM	0.0652	0.0184	0.0224	0.0339	0.0186

Among no-DP synthesizers, CTGAN outperforms most methods in terms of privacy protection, aligning with the expectation that models with lower fidelity generally provide stronger privacy guarantees. Additionally, statistical methods such as LDM demonstrate

stable privacy performance, suggesting that their structured data synthesis approach inherently reduces membership leakage risks. However, deep generative models such as TAVE (0.1211 on TON, 0.0155 on UGR16) and GReaT (0.0346 on TON, 0.0148 on UGR16) exhibit relatively higher MDS scores, indicating that while these models achieve superior fidelity, they are more vulnerable to membership inference attacks.

For DP synthesizers, NetDPSyn provides the strongest privacy protection across all datasets, with MDS scores consistently remaining low, such as 0.0068 on TON and 0.0028 on UGR16, demonstrating the effectiveness of its DP mechanism in mitigating privacy leakage risks. As shown in Figure 2, NetDPSyn outperforms NetShare in terms of both privacy protection (MDS) and classification accuracy. Moreover, its classification accuracy increases with a higher privacy budget ϵ, further indicating the superior quality of its synthetic data. In contrast, NetShare fails to match NetDPSyn's classification accuracy even under a high ϵ, suggesting potential limitations in its data synthesis approach. While DPNetSynDM exhibits slightly higher MDS values compared to NetDPSyn, it still provides stronger privacy protection than no-DP synthesizers, with MDS scores of 0.0652 on TON and 0.0186 on UGR16, indicating that it remains a viable choice for privacy-preserving data synthesis.

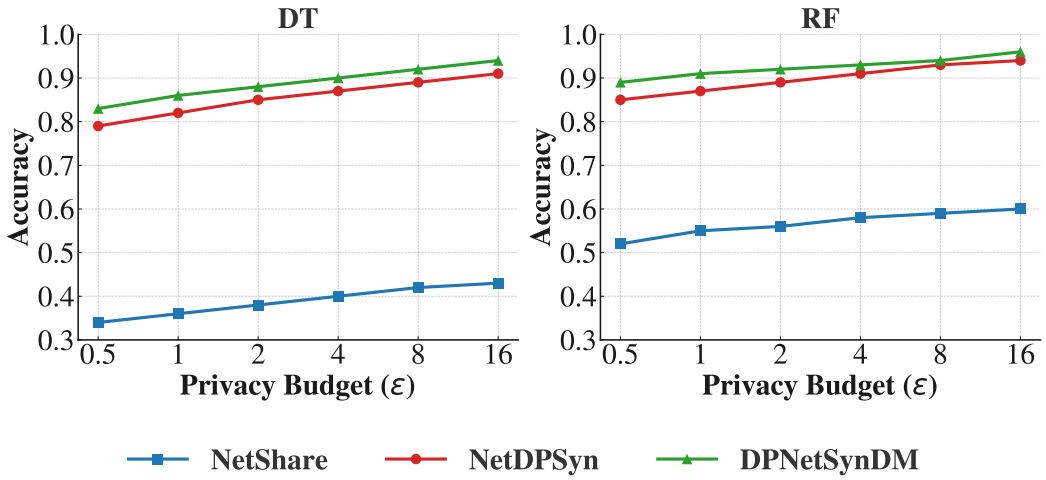

Figure 2. A comparison of TON (NetFlow) accuracy between NetDPSyn and NetShare with a large range of ϵ.

Overall, the results further confirm the complex trade-off between privacy protection and data utility. NetDPSyn effectively reduces privacy risks but do so at the cost of fidelity. Conversely, TAVE and NetSynDM generate higher-quality synthetic data but are more susceptible to privacy attacks. Future research should focus on optimizing the noise injection mechanism in DPNetSynDM to minimize the impact of differential privacy on data quality, thereby achieving a better balance between fidelity and privacy protection.

As illustrated in Figure 3, we present a comprehensive histogram comparing the privacy performances of various synthesizers across multiple datasets. The histogram clearly highlights the superiority of DP synthesizers, specifically NetDPSyn and DPNetSynDM, in providing robust privacy protection. NetDPSyn consistently exhibits the lowest membership disclosure scores (MDSs), affirming its effectiveness in maintaining extremely low privacy leakage risks. Notably, DPNetSynDM, while slightly behind NetDPSyn, significantly outperforms traditional no-DP synthesizers such as CTGAN, TAVE, GReat, and LDM. Its consistently low MDS values across all datasets further demonstrate the benefits of incorporating differential privacy into diffusion-based models. This visual representation underscores the effectiveness of DP methods in minimizing membership inference risks,

thereby providing empirical evidence supporting DPNetSynDM's position as a highly viable approach for privacy-preserving data synthesis.

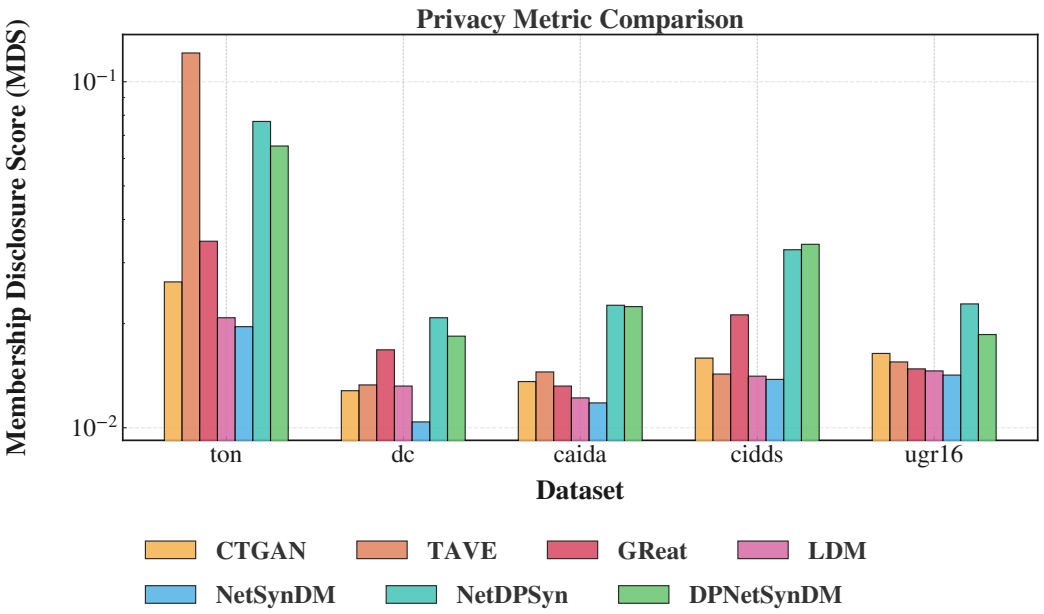

Figure 3. Privacy comparison across datasets.

6.3. Utility Evaluation

We evaluate the effectiveness of data synthesis by applying downstream tasks to synthetic datasets and measuring the results using our defined metrics, including machine learning affinity (MLA) and query error, as reported in Tables 5 and 6. The results highlight key differences between no-DP synthesizers and DP synthesizers in terms of utility preservation.

Table 5. Machine learning affinity of data synthesis is evaluated, where a lower value indicates that the performance of synthetic data is more similar to that of real data. The privacy budget ϵ of no-DP synthesizers is set to ∞ (top), and the budget for differential privacy (DP) synthesizers is set to 1 (bottom).

Methods	TON	DC	CAIDA	CIDDS	UGR16
CTGAN	0.000265 ± 0.000122	0.580068 ± 0.025935	0.530987 ± 0.014147	0.000100 ± 0.000035	0.297094 ± 0.005937
TAVE	0.000797 ± 0.000188	0.702786 ± 0.023150	0.475315 ± 0.012104	0.001135 ± 0.001017	0.304603 ± 0.095859
GReat	0.085016 ± 0.002554	0.082346 ± 0.002658	0.084020 ± 0.002714	0.087224 ± 0.002212	0.08344 ± 0.0024
LDM	0.000018 ± 0.000012	0.108598 ± 0.004408	0.076747 ± 0.003395	0.000009 ± 0.000004	0.1230 ± 0.1063
NetSynDM	0.000004 ± 0.000004	0.222335 ± 0.009213	2.207452 ± 0.365110	0.000003 ± 0.000002	0.2676 ± 0.1254
NetDPSyn	0.000378 ± 0.000777	1.512826 ± 0.148622	1.234690 ± 0.143961	0.005015 ± 0.000815	0.005416 ± 0.000518
DPNetSynDM	0.000331 ± 0.009779	1.422954 ± 0.112818	1.357546 ± 0.094102	0.004411 ± 0.006152	0.003554 ± 0.000216

Table 6. Query error evaluation of data synthesis, where a lower value indicates a smaller query error. The privacy budget ϵ of no-DP synthesizers is set to ∞ (top), and the budget for DP synthesizers is set to 1 (bottom).

Methods	TON	DC	CAIDA	CIDDS	UGR16
CTGAN	0.026149 ± 0.002224	0.012809 ± 0.001444	0.021641 ± 0.002198	0.044259 ± 0.004962	0.031250 ± 0.004651
TAVE	0.008266 ± 0.001252	0.017962 ± 0.001891	0.009775 ± 0.001189	0.004328 ± 0.000575	0.011067 ± 0.006042
GReat	0.009468 ± 0.000664	0.005451 ± 0.000432	0.010898 ± 0.000810	0.034348 ± 0.000426	0.006684 ± 0.002112
LDM	0.007130 ± 0.000951	0.003884 ± 0.000326	0.004022 ± 0.000430	0.002174 ± 0.000317	0.0039 ± 0.0023
NetSynDM	0.003689 ± 0.000390	0.003487 ± 0.000342	0.082973 ± 0.009303	0.002096 ± 0.000162	0.0036 ± 0.0019
NetDPSyn	0.009992 ± 0.001142	0.003223 ± 0.004327	0.053591 ± 0.010868	0.028328 ± 0.004384	0.0032 ± 0.0016
DPNetSynDM	0.007382 ± 0.004230	0.008117 ± 0.004310	0.041462 ± 0.005987	0.022666 ± 0.004780	0.0034 ± 0.0021

For tasks related to machine learning, the best performance among no-DP synthesizers is achieved by CTGAN on the UGR16 dataset, showing the highest machine learning affinity. However, deep generative models such as TAVE and GReat demonstrate stronger generalization in other datasets, particularly in complex traffic datasets like CAIDA. Meanwhile, statistical methods such as LDM maintain stable utility performance across various datasets, though they do not always outperform deep generative models in complex settings.

When privacy constraints are introduced, DP synthesizers exhibit a clear performance drop in machine learning affinity. NetDPSyn outperforms DPNetSynDM in most cases, suggesting that its privacy mechanism is more optimized for retaining utility while ensuring privacy. DPNetSynDM, in contrast, suffers from a larger loss in machine learning affinity, likely due to its noise injection method affecting learned representations more significantly.

For query error evaluation, the results in Table 6 reinforce findings from fidelity assessments. NetDPSyn achieves the lowest query errors among DP synthesizers, indicating its robustness in preserving statistical properties even under DP constraints. Meanwhile, TAVE and NetSynDM maintain strong performances among no-DP synthesizers, showing lower query errors compared to deep generative models such as GReat. Interestingly, DPNetSynDM exhibits notably higher query errors, reinforcing the challenge of balancing differential privacy with query accuracy.

Overall, the results suggest that while no-DP synthesizers (e.g., TAVE, NetSynDM) provide superior utility, they lack privacy guarantees. DP synthesizers (NetDPSyn) offer a better balance between privacy and utility, but noise-based approaches such as DPNetSynDM require further optimization to enhance their performances. Future work should explore hybrid strategies that can optimize both machine learning affinity and query accuracy under differential privacy constraints.

7. Discussion

NetSynDM and DPNetSynDM, as two different data synthesis methods, exhibit significant differences in terms of utility and privacy protection. NetSynDM performs well among no-DP synthesizers, particularly in the TON, DC, and CAIDA datasets, where its Wasserstein distance remains low, indicating that the synthetic data distribution closely resembles that of the real data. However, DPNetSynDM shows higher Wasserstein distances, especially in the CIDDS dataset, suggesting that its fidelity is affected by the noise introduced by the DP mechanism, leading to a certain degree of data quality degradation. Nevertheless, DPNetSynDM maintains reasonable fidelity across multiple datasets, demonstrating its balance between privacy protection and data quality.

In the evaluation of machine learning affinity, NetSynDM performs exceptionally well in no-DP settings, effectively mimicking real data distributions. However, when the differential privacy mechanism is introduced, DPNetSynDM exhibits a noticeable decline in machine learning affinity, indicating that the alignment between its synthetic and real data is impacted. This reduction in utility can be attributed to the noise introduced during the data generation process by the DP mechanism, which disrupts certain data patterns, leading to suboptimal performance in machine learning tasks compared to no-DP synthesizers. From a privacy perspective, however, this reduction in utility is acceptable, as DPNetSynDM ensures that synthetic data does not leak sensitive information from the original dataset, making it more competitive in privacy-prioritized applications.

Regarding query error evaluation, NetSynDM achieves lower query errors under no-DP settings, indicating its robustness in preserving statistical relationships, while DPNetSynDM exhibits relatively higher query errors, suggesting that its privacy mechanism impacts data consistency to some extent. Despite this, DPNetSynDM maintains a certain level of statistical integrity while ensuring privacy protection, making it more advantageous

in scenarios where data sharing and strict differential privacy requirements are necessary. Unlike NetSynDM, which focuses solely on data fidelity, DPNetSynDM provides rigorous mathematical privacy guarantees, ensuring that synthetic data cannot be exploited for membership inference attacks or other privacy breaches, making it particularly suitable for high-privacy applications such as healthcare, finance, and government data sharing.

It is worth noting that another differentially private synthesizer, NetDPSyn, theoretically achieves the strongest possible privacy guarantee (e.g., an MDS score of 0). However, its data quality may be severely impacted due to the hard-injection differential privacy mechanism employed, which tends to introduce excessive noise, disrupting the original data distribution and patterns, thus significantly degrading data utility. In comparison, while DPNetSynDM does not reach the theoretical maximum privacy guarantee of NetDPSyn, it achieves a more balanced trade-off between data utility and privacy protection, making it more suitable for practical application scenarios.

Although DPNetSynDM has some limitations in fidelity and query accuracy, its strong privacy protection capabilities make it significantly advantageous in environments requiring strict data security. The current limitations primarily stem from the impact of its privacy noise strategy, which can be optimized in the future to reduce unnecessary data distortions and improve fidelity. Additionally, adaptive DP mechanisms can allocate privacy budgets based on different data characteristics, thereby minimizing the utility loss caused by privacy protection. A hybrid approach combining statistical and deep learning methods is also a viable optimization direction, allowing for a better balance between data fidelity and privacy protection. Furthermore, DPNetSynDM has potential applications in federated learning and privacy-preserving computation, ensuring that data remains strictly protected while being shared across different institutions.

In summary, NetSynDM is best suited for applications that prioritize high data fidelity with minimal privacy constraints, such as internal model pretraining, anomaly detection simulation, or stress testing in secure environments. Conversely, DPNetSynDM is preferable in privacy-critical scenarios, including public data releases or cross-institutional collaboration in sensitive domains such as healthcare and finance, where a certain degree of fidelity loss is acceptable in exchange for formal privacy guarantees. This comparison underscores the inherent trade-off between utility and privacy. Future research may focus on mitigating this gap through adaptive differential privacy strategies or hybrid synthesis approaches that balance fidelity and privacy more effectively.

8. Related Works

Currently, various works state that advanced machine learning systems face huge challenges of security concerns, calling for more security protection to promote the application of AI systems in reality [56–58].

8.1. Network Data Synthesis

Network data synthesis aims to produce synthetic data that can mimic real network properties, with important uses in privacy protection and model training. As a result, generative models for network data have become increasingly important and have broad applications [59–61]. To address the problem of imbalanced categorical features, Xu et al. [19] proposed CTGAN and TVAE, which build on GAN [62] and VAE [63] techniques.

Beyond these, a variety of advanced methods for creating synthetic data have also been explored. In particular, TVAE [19] stands out as a prominent VAE-based synthesizer tailored for tabular data generation. It adopts a normalization strategy that is sensitive to the specific modes of data distribution, effectively addressing the challenges posed by non-Gaussian continuous variables. GOGGLE [64] is another VAE-based approach that

uses graph neural networks as both the encoder and decoder. Motivated by the success of large language models in capturing natural language distributions, GReaT [20] adopts the autoregressive GPT2 model to learn sentence-level distributions. It represents records as textual sequences for the language model and generates synthetic data using prompts.

8.2. Privacy Leakage for Network Dataset

One main approach for addressing privacy leakage in datasets is to create synthetic data that retains the statistical properties of real data while protecting sensitive information. Recent studies show that MDS can effectively detect privacy risks across different synthesizers, revealing that GReaT, in particular, faces a high risk of member leakage. By applying differential privacy to the synthesizer, this risk can be significantly reduced [46].

8.3. DP Dataset Synthesis

Differentially private (DP) dataset synthesis is commonly employed to generate synthetic datasets that can be securely distributed under strong privacy guarantees [65]. In recent years, numerous DP-based generation methods have emerged across a wide range of data domains, including images [31,66–69], tabular data [49,70–74], graphs [75], time-series [76], trajectories [77,78], and text [79,80], among others. DP ensures that the output of any analysis on a dataset, including queries, will not significantly differ whether a particular individual's data is part of the dataset or not.

8.4. DP for Network Communication

The application of DP in network communication is being increasingly widely studied, especially for protecting data privacy, improving communication security, and reducing privacy disclosure. Our survey reveals that recent studies have frequently adopted generative models to synthesize network communication data. Among them, Generative Adversarial Networks (GANs) [15,81–87] are predominantly used. However, these methods often fall short in providing strong privacy protection for network traces. Stadler et al. [88] point out that models like CTGAN [19] are vulnerable to linkage attacks, enabling adversaries to confidently identify whether specific records exist in the original dataset.

9. Conclusions

Network trace synthesis plays a crucial role in enabling data-driven network analysis while ensuring privacy protection. Our evaluation of different synthesizers reveals clear trade-offs between fidelity, privacy, and utility, highlighting the strengths and weaknesses of various approaches. By systematically analyzing fidelity, machine learning affinity, and query errors across multiple datasets, we provide insights into the capabilities of no-DP synthesizers and differentially private (DP) synthesizers, offering valuable guidance for practitioners.

Our findings highlight that NetSynDM achieves strong fidelity and utility, making it a suitable choice for scenarios where preserving the original data distribution is a priority. With particularly low Wasserstein distances in datasets such as `TON`, `DC`, and `CAIDA`, it demonstrates effective fidelity across both temporal and distributional aspects. However, it lacks formal privacy guarantees, making it less suitable for applications requiring strong data protection. DPNetSynDM, on the other hand, ensures strict privacy protection through differential privacy (DP) mechanisms, albeit at the cost of higher query errors and reduced machine learning affinity. Despite these limitations, DPNetSynDM remains a viable solution for privacy-sensitive applications, particularly in regulated environments such as healthcare, finance, and secure data sharing.

Several key insights emerge from our study:

- Hyperparameter tuning is critical for maximizing the performance of both no-DP and DP synthesizers. Optimizing model configurations can substantially improve fidelity and reduce query errors, particularly in deep generative models.
- Statistical methods exhibit stable privacy performance, making them preferable for applications where privacy is the highest priority. LDM, for instance, achieves consistent utility with low membership disclosure risks, positioning it as a reliable choice for privacy-focused settings.
- Diffusion models strike a balance between fidelity and privacy, with DPNetSynDM demonstrating the potential of integrating DP mechanisms with diffusion-based synthesis. While its fidelity lags behind those of no-DP models, its strong privacy guarantees make it a promising candidate for secure data synthesis.
- Deep generative models offer flexibility and adaptability for task-specific applications. Methods such as TAVE and GReaT show superior generalization capabilities, making them well-suited for machine learning-driven network analysis and other data-intensive tasks.

In light of these findings, we identify several promising directions for future research:

- Adaptive privacy budget allocation: Future research may explore dynamically assigning differential privacy budgets based on feature sensitivity or task-specific utility requirements. This approach aims to preserve data fidelity while maintaining strong privacy guarantees.
- Domain-specific synthesis strategies: Tailoring network trace generation methods for specific domains, such as healthcare, industrial control systems, or the Internet of Things (IoT), can leverage domain knowledge and specialized data structures to enhance synthesis quality and downstream performance.
- Cross-institution collaboration mechanisms: Facilitating collaborative analysis and federated learning across institutions using DP-synthesized data—without sharing raw data—remains a key challenge worthy of further investigation.

Overall, our study highlights the dynamic progress made in network trace synthesis, where recent advances in deep generative and diffusion models bring new capabilities and challenges. Bridging the gap between no-DP and DP synthesizers—where privacy often comes at the cost of fidelity—remains an open problem. We hope that our systematic evaluation framework can serve as a foundation for future developments, guiding the community toward more secure, useful, and practical data synthesis solutions.

Author Contributions: Conceptualization, J.-X.Y. and Y.-H.X.; methodology, J.-X.Y.; software, J.-X.Y.; validation, J.-X.Y. and Y.-H.X.; formal analysis, J.-X.Y.; investigation, J.-X.Y.; resources, M.H. and G.Y.; data curation, W.Z.; writing—original draft preparation, J.-X.Y.; writing—review and editing, Y.-H.X. and J.-X.Y.; visualization, J.-X.Y.; supervision, Y.-H.X.; project administration, Y.-H.X.; funding acquisition, Y.-H.X. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant 61601275 and in part by the Jiangsu Graduate Research and Practice Innovation Program under Grant SJCX24_0385.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset used and analyzed during the current study is available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Jiang, X.; Liu, S.; Gember-Jacobson, A.; Schmitt, P.; Bronzino, F.; Feamster, N. Generative, high-fidelity network traces. In Proceedings of the 22nd ACM Workshop on Hot Topics in Networks, Cambridge, MA, USA, 28–29 November 2023; pp. 131–138.
2. Lyu, M.; Gharakheili, H.H.; Sivaraman, V. A survey on enterprise network security: Asset behavioral monitoring and distributed attack detection. *IEEE Access* **2024**, *12*, 89363–89383. [CrossRef]
3. Fuentes-García, M.; Camacho, J.; Maciá-Fernández, G. Present and future of network security monitoring. *IEEE Access* **2021**, *9*, 112744–112760. [CrossRef]
4. Devan, M.; Shanmugam, L.; Tomar, M. AI-powered data migration strategies for cloud environments: Techniques, frameworks, and real-world applications. *Aust. J. Mach. Learn. Res. Appl.* **2021**, *1*, 79–111.
5. Kenyon, A.; Deka, L.; Elizondo, D. Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets. *Comput. Secur.* **2020**, *99*, 102022. [CrossRef]
6. Sun, D.; Chen, J.Q.; Gong, C.; Wang, T.; Li, Z. NetDPSyn: Synthesizing Network Traces under Differential Privacy. In Proceedings of the 2024 ACM on Internet Measurement Conference, Madrid, Spain, 4–6 November 2024; pp. 545–554.
7. Ghalebikesabi, S.; Wilde, H.; Jewson, J.; Doucet, A.; Vollmer, S.; Holmes, C. Mitigating statistical bias within differentially private synthetic data. In Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence, Eindhoven, The Netherlands, 1–5 August 2022; PMLR: New York, NY, USA, 2022; pp. 696–705.
8. Wei, C.; Li, W.; Chen, G.; Chen, W. DC-SGD: Differentially Private SGD with Dynamic Clipping through Gradient Norm Distribution Estimation. *arXiv* **2025**, arXiv:2503.22988. [CrossRef]
9. Novado, D.; Cohen, E.; Foster, J. Multi-tier privacy protection for large language models using differential privacy. *Authorea* **2024**. [CrossRef]
10. Keshun, Y.; Guangqi, Q.; Yingkui, G. Optimizing prior distribution parameters for probabilistic prediction of remaining useful life using deep learning. *Reliab. Eng. Syst. Saf.* **2024**, *242*, 109793. [CrossRef]
11. Saxena, D.; Cao, J. Generative adversarial networks (GANs) challenges, solutions, and future directions. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 63.
12. Navidan, H.; Moshiri, P.F.; Nabati, M.; Shahbazian, R.; Ghorashi, S.A.; Shah-Mansouri, V.; Windridge, D. Generative Adversarial Networks (GANs) in networking: A comprehensive survey & evaluation. *Comput. Netw.* **2021**, *194*, 108149.
13. Akkem, Y.; Biswas, S.K.; Varanasi, A. A comprehensive review of synthetic data generation in smart farming by using variational autoencoder and generative adversarial network. *Eng. Appl. Artif. Intell.* **2024**, *131*, 107881. [CrossRef]
14. Liang, S.; Pan, Z.; Liu, W.; Yin, J.; de Rijke, M. A Survey on Variational Autoencoders in Recommender Systems. *ACM Comput. Surv.* **2024**, *56*, 268. [CrossRef]
15. Yin, Y.; Lin, Z.; Jin, M.; Fanti, G.; Sekar, V. Practical gan-based synthetic ip header trace generation using netshare. In Proceedings of the ACM SIGCOMM 2022 Conference, Amsterdam, The Netherlands, 22–26 August 2022; pp. 458–472.
16. Yang, Z.; Zhan, F.; Liu, K.; Xu, M.; Lu, S. Ai-generated images as data source: The dawn of synthetic era. *arXiv* **2023**, arXiv:2310.01830. [CrossRef]
17. Liu, M.F.; Lyu, S.; Vinaroz, M.; Park, M. Differentially private latent diffusion models. *arXiv* **2023**, arXiv:2305.15759.
18. Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H.B.; Mironov, I.; Talwar, K.; Zhang, L. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 308–318.
19. Xu, L.; Skoularidou, M.; Cuesta-Infante, A.; Veeramachaneni, K. Modeling tabular data using conditional gan. In Proceedings of the Advances in Neural Information Processing Systems 32 (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019.
20. Borisov, V.; Seßler, K.; Leemann, T.; Pawelczyk, M.; Kasneci, G. Language models are realistic tabular data generators. *arXiv* **2022**, arXiv:2210.06280.
21. Bagnulo, M.; Matthews, P.; van Beijnum, I. *Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers*; Technical Report; RFC Editor: Marina del Rey, CA, USA, 2011.
22. Gruteser, M.; Grunwald, D. Enhancing location privacy in wireless LAN through disposable interface identifiers: A quantitative analysis. In Proceedings of the 1st ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots, San Diego, CA, USA, 19 September 2003; pp. 46–55.
23. Jain, S.; Javed, M.; Paxson, V. Towards mining latent client identifiers from network traffic. *Proc. Priv. Enhancing Technol.* **2016**, *2016*, 100–114. [CrossRef]
24. Xu, J.; Fan, J.; Ammar, M.H.; Moon, S.B. Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In Proceedings of the 10th IEEE International Conference on Network Protocols, Paris, France, 12–15 November 2002; IEEE: Piscataway, NJ, USA, 2002; pp. 280–289.
25. Imana, B.; Korolova, A.; Heidemann, J. Institutional privacy risks in sharing DNS data. In Proceedings of the Applied Networking Research Workshop, Online, 24–30 July 2021; pp. 69–75.

26. Sohl-Dickstein, J.; Weiss, E.; Maheswaranathan, N.; Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; PMLR: New York, NY, USA, 2015; pp. 2256–2265.
27. Ho, J.; Jain, A.; Abbeel, P. Denoising diffusion probabilistic models. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 6840–6851.
28. Luo, C. Understanding diffusion models: A unified perspective. *arXiv* **2022**, arXiv:2208.11970. [CrossRef]
29. Croitoru, F.A.; Hondru, V.; Ionescu, R.T.; Shah, M. Diffusion models in vision: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 10850–10869. [CrossRef]
30. Yang, L.; Zhang, Z.; Song, Y.; Hong, S.; Xu, R.; Zhao, Y.; Zhang, W.; Cui, B.; Yang, M.H. Diffusion models: A comprehensive survey of methods and applications. *ACM Comput. Surv.* **2023**, *56*, 105. [CrossRef]
31. Li, K.; Gong, C.; Li, Z.; Zhao, Y.; Hou, X.; Wang, T. PrivImage: Differentially Private Synthetic Image Generation using Diffusion Models with Semantic-Aware Pretraining. In Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24), Philadelphia, PA, USA, 14–16 August 2024; pp. 4837–4854.
32. Jiang, X.; Liu, S.; Gember-Jacobson, A.; Bhagoji, A.N.; Schmitt, P.; Bronzino, F.; Feamster, N. Netdiffusion: Network data augmentation through protocol-constrained traffic generation. *Proc. ACM Meas. Anal. Comput. Syst.* **2024**, *8*, 11. [CrossRef]
33. Zhang, S.; Li, T.; Jin, D.; Li, Y. NetDiff: A service-guided hierarchical diffusion model for network flow trace generation. *Proc. ACM Netw.* **2024**, *2*, 16. [CrossRef]
34. Chai, H.; Jiang, T.; Yu, L. Diffusion model-based mobile traffic generation with open data for network planning and optimization. In Proceedings of the KDD'24, Barcelona, Spain, 25–29 August 2024; pp. 4828–4838. [CrossRef]
35. Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; Ommer, B. High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 28–24 June 2022; pp. 10684–10695.
36. Dwork, C.; Roth, A. The algorithmic foundations of differential privacy. *Found. Trends® Theor. Comput. Sci.* **2014**, *9*, 211–407. [CrossRef]
37. Dwork, C.; McSherry, F.; Nissim, K.; Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography, Proceedings of the Third Theory of Cryptography Conference, New York, NY, USA, 4–7 March 2006*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 265–284.
38. Song, S.; Chaudhuri, K.; Sarwate, A.D. Stochastic gradient descent with differentially private updates. In Proceedings of the 2013 IEEE Global Conference on Signal and Information Processing, Austin, TX, USA, 3–5 December 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 245–248.
39. Du, J.; Li, S.; Chen, X.; Chen, S.; Hong, M. Dynamic differential-privacy preserving sgd. *arXiv* **2021**, arXiv:2111.00173.
40. Yousefpour, A.; Shilov, I.; Sablayrolles, A.; Testuggine, D.; Prasad, K.; Malek, M.; Nguyen, J.; Ghosh, S.; Bharadwaj, A.; Zhao, J.; et al. Opacus: User-friendly differential privacy library in PyTorch. *arXiv* **2021**, arXiv:2109.12298.
41. Torfi, A.; Fox, E.A.; Reddy, C.K. Differentially private synthetic medical data generation using convolutional GANs. *Inf. Sci.* **2022**, *586*, 485–500. [CrossRef]
42. Mironov, I. Rényi differential privacy. In Proceedings of the 2017 IEEE 30th Computer Security Foundations Symposium (CSF), Santa Barbara, CA, USA, 21–25 August 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 263–275.
43. Zhang, Y.; Wen, J.; Yang, G.; He, Z.; Wang, J. Path loss prediction based on machine learning: Principle, method, and data expansion. *Appl. Sci.* **2019**, *9*, 1908. [CrossRef]
44. Wang, J.; Liu, S.; Li, Y. A review of differential privacy in individual data release. *Int. J. Distrib. Sens. Netw.* **2015**, *11*, 259682. [CrossRef]
45. Abay, N.C.; Zhou, Y.; Kantarcioglu, M.; Thuraisingham, B.; Sweeney, L. Privacy preserving synthetic data release using deep learning. In Proceedings of the Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, 10–14 September 2018; Springer: Cham, Switzerland, 2019; pp. 510–526.
46. Du, Y.; Li, N. Towards principled assessment of tabular data synthesis algorithms. *arXiv* **2024**, arXiv:2402.06806.
47. Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 2623–2631.
48. McKenna, R.; Miklau, G.; Sheldon, D. Winning the nist contest: A scalable and general approach to differentially private synthetic data. *arXiv* **2021**, arXiv:2108.04978. [CrossRef]
49. Cai, K.; Lei, X.; Wei, J.; Xiao, X. Data synthesis via differentially private markov random fields. *Proc. VLDB Endow.* **2021**, *14*, 2190–2202. [CrossRef]
50. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A.C. Improved training of wasserstein gans. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
51. Flamary, R.; Courty, N.; Gramfort, A.; Alaya, M.Z.; Boisbunon, A.; Chambon, S.; Chapel, L.; Corenflos, A.; Fatras, K.; Fournier, N.; et al. Pot: Python optimal transport. *J. Mach. Learn. Res.* **2021**, *22*, 1–8.

52. El Emam, K.; Mosquera, L.; Fang, X. Validating a membership disclosure metric for synthetic health data. *JAMIA Open* **2022**, *5*, ooac083. [CrossRef] [PubMed]
53. Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership Inference Attacks Against Machine Learning Models. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–24 May 2017; pp. 3–18. [CrossRef]
54. Prokhorenkova, L.; Gusev, G.; Vorobev, A.; Dorogush, A.V.; Gulin, A. CatBoost: Unbiased boosting with categorical features. In Proceedings of the Advances in Neural Information Processing Systems 31 (NeurIPS 2018), Montreal, QC, Canada, 3–8 December 2018.
55. Gorishniy, Y.; Rubachev, I.; Khrulkov, V.; Babenko, A. Revisiting deep learning models for tabular data. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 18932–18943.
56. Gong, C.; Yang, Z.; Bai, Y.; He, J.; Shi, J.; Li, K.; Sinha, A.; Xu, B.; Hou, X.; Lo, D.; et al. Baffle: Hiding backdoors in offline reinforcement learning datasets. In Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2024; pp. 2086–2104.
57. Gong, C.; Yang, Z.; Bai, Y.; Shi, J.; Sinha, A.; Xu, B.; Lo, D.; Hou, X.; Fan, G. Curiosity-driven and victim-aware adversarial policies. In Proceedings of the 38th Annual Computer Security Applications Conference, Austin, TX, USA, 5–9 December 2022; pp. 186–200.
58. Gong, C.; Li, K.; Yao, J.; Wang, T. Trajdeleter: Enabling trajectory forgetting in offline reinforcement learning agents. *arXiv* **2024**, arXiv:2404.12530. [CrossRef]
59. Assefa, S.A.; Dervovic, D.; Mahfouz, M.; Tillman, R.E.; Reddy, P.; Veloso, M. Generating synthetic data in finance: Opportunities, challenges and pitfalls. In Proceedings of the First ACM International Conference on AI in Finance, New York, NY, USA, 15–16 October 2020; pp. 1–8.
60. Zheng, S.; Charoenphakdee, N. Diffusion models for missing value imputation in tabular data. *arXiv* **2022**, arXiv:2210.17128.
61. Hernandez, M.; Epelde, G.; Alberdi, A.; Cilla, R.; Rankin, D. Synthetic data generation for tabular health records: A systematic review. *Neurocomputing* **2022**, *493*, 28–45. [CrossRef]
62. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems 27 (NIPS 2014), Montreal, QC, Canada, 8–13 December 2014.
63. Kingma, D.P. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
64. Liu, T.; Qian, Z.; Berrevoets, J.; van der Schaar, M. GOGGLE: Generative modelling for tabular data by learning relational structure. In Proceedings of the the Eleventh International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
65. Hu, Y.; Wu, F.; Li, Q.; Long, Y.; Garrido, G.M.; Ge, C.; Ding, B.; Forsyth, D.; Li, B.; Song, D. Sok: Privacy-preserving data synthesis. In Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 4696–4713.
66. Dockhorn, T.; Cao, T.; Vahdat, A.; Kreis, K. Differentially private diffusion models. *arXiv* **2022**, arXiv:2210.09929.
67. Liew, S.P.; Takahashi, T.; Ueno, M. Pearl: Data synthesis via private embeddings and adversarial reconstruction learning. *arXiv* **2021**, arXiv:2106.04590.
68. Gong, C.; Li, K.; Lin, Z.; Wang, T. DPImageBench: A Unified Benchmark for Differentially Private Image Synthesis. *arXiv* **2025**, arXiv:2503.14681.
69. Li, K.; Gong, C.; Li, X.; Zhao, Y.; Hou, X.; Wang, T. From easy to hard: Building a shortcut for differentially private image synthesis. In Proceedings of the 2025 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 12–15 May 2025; IEEE: Piscataway, NJ, USA, 2025; pp. 3988–4006.
70. Cai, K.; Xiao, X.; Cormode, G. Privlava: Synthesizing relational data with foreign keys under differential privacy. *Proc. ACM Manag. Data* **2023**, *1*, 142. [CrossRef]
71. McKenna, R.; Mullins, B.; Sheldon, D.; Miklau, G. Aim: An adaptive and iterative mechanism for differentially private synthetic data. *arXiv* **2022**, arXiv:2201.12677. [CrossRef]
72. Jordon, J.; Yoon, J.; Van Der Schaar, M. PATE-GAN: Generating synthetic data with differential privacy guarantees. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
73. Zhang, J.; Cormode, G.; Procopiuc, C.M.; Srivastava, D.; Xiao, X. Privbayes: Private data release via bayesian networks. *ACM Trans. Database Syst. (TODS)* **2017**, *42*, 25. [CrossRef]
74. Zhang, Z.; Wang, T.; Li, N.; Honorio, J.; Backes, M.; He, S.; Chen, J.; Zhang, Y. PrivSyn: Differentially private data synthesis. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), Vancouver, BC, Canada, 11–13 August 2021; pp. 929–946.
75. Yuan, Q.; Zhang, Z.; Du, L.; Chen, M.; Cheng, P.; Sun, M. PrivGraph: Differentially Private Graph Data Publication by Exploiting Community Information. In Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23), Anaheim, CA, USA, 9–11 August 2023; pp. 3241–3258.

76. Frigerio, L.; de Oliveira, A.S.; Gomez, L.; Duverger, P. Differentially private generative adversarial networks for time series, continuous, and discrete open data. In Proceedings of the ICT Systems Security and Privacy Protection: 34th IFIP TC 11 International Conference, SEC 2019, Lisbon, Portugal, 25–27 June 2019; Springer: Cham, Switzerland, 2019; pp. 151–164.
77. Du, Y.; Hu, Y.; Zhang, Z.; Fang, Z.; Chen, L.; Zheng, B.; Gao, Y. Ldptrace: Locally differentially private trajectory synthesis. *Proc. VLDB Endow.* **2023**, *16*, 1897–1909. [CrossRef]
78. Wang, H.; Zhang, Z.; Wang, T.; He, S.; Backes, M.; Chen, J.; Zhang, Y. PrivTrace: Differentially Private Trajectory Synthesis by Adaptive Markov Models. In Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23), Anaheim, CA, USA, 9–11 August 2023; pp. 1649–1666.
79. Tang, X.; Shin, R.; Inan, H.A.; Manoel, A.; Mireshghallah, F.; Lin, Z.; Gopi, S.; Kulkarni, J.; Sim, R. Privacy-preserving in-context learning with differentially private few-shot generation. *arXiv* **2023**, arXiv:2309.11765.
80. Yue, X.; Inan, H.A.; Li, X.; Kumar, G.; McAnallen, J.; Shajari, H.; Sun, H.; Levitan, D.; Sim, R. Synthetic text generation with differential privacy: A simple and practical recipe. *arXiv* **2022**, arXiv:2210.14348.
81. Cheng, A. PAC-GAN: Packet generation of network traffic using generative adversarial networks. In Proceedings of the 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 17–19 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 728–734.
82. Fan, L.; Pokkunuru, A. DPNeT: Differentially private network traffic synthesis with generative adversarial networks. In Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy, Calgary, AB, Canada, 19–20 July 2021; Springer: Cham, Switzerland, 2021; pp. 3–21.
83. Han, L.; Sheng, Y.; Zeng, X. A packet-length-adjustable attention model based on bytes embedding using flow-wgan for smart cybersecurity. *IEEE Access* **2019**, *7*, 82913–82926. [CrossRef]
84. Lin, Z.; Jain, A.; Wang, C.; Fanti, G.; Sekar, V. Using gans for sharing networked time series data: Challenges, initial promise, and open questions. In Proceedings of the ACM Internet Measurement Conference, Online, 27–29 October 2020; pp. 464–483.
85. Ring, M.; Schlör, D.; Landes, D.; Hotho, A. Flow-based network traffic generation using generative adversarial networks. *Comput. Secur.* **2019**, *82*, 156–172. [CrossRef]
86. Sivaroopan, N.; Madarasingha, C.; Muramudalige, S.; Jourjon, G.; Jayasumana, A.; Thilakarathna, K. SyNIG: Synthetic network traffic generation through time series imaging. In Proceedings of the 2023 IEEE 48th Conference on Local Computer Networks (LCN), Daytona Beach, FL, USA, 2–5 October 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 1–9.
87. Wang, P.; Li, S.; Ye, F.; Wang, Z.; Zhang, M. PacketCGAN: Exploratory study of class imbalance for encrypted traffic classification using CGAN. In Proceedings of the ICC 2020—2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–7.
88. Stadler, T.; Oprisanu, B.; Troncoso, C. Synthetic data–anonymisation groundhog day. In Proceedings of the 31st USENIX Security Symposium (USENIX Security 22), Boston, MA, USA, 10–12 August 2022; pp. 1451–1468.