Bulletin of the Technical Community on

Data Engineering

June 2023 Vol. 47 No. 2

IEEE Computer Society

Letters

The TC on Data Engineering
Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems. The TCDE web page is http://tab.computer.org/tcde/index.html.

The Data Engineering Bulletin
The Bulletin of the Technical Community on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

The Data Engineering Bulletin web site is at http://tab.computer.org/tcde/bull_about.html.

Letter from the Editor-in-Chief

As the world becomes increasingly data-driven, the need for Privacy-Preserving Data Management has never been more crucial. This task goes beyond simple data protection, aiming to secure sensitive information while enabling its beneficial uses in research, analytics, and beyond. Central to this mission is the integration of advanced technical solutions with robust governance practices, recognizing that data quality and security are vital to the trustworthiness and effectiveness of AI systems.

In this June edition of the Data Engineering Bulletin, we focus on the complex realm of privacy-preserving data management. This issue, meticulously curated by Xiaokui Xiao, features six pioneering papers that expand our current understanding and capabilities.

For example, Gavidia-Calderon et al. introduce SQLSynthGen, a tool that tackles the dual challenges of data fidelity and privacy. This innovative method for generating synthetic data ensures that datasets maintain the statistical properties and utility of real-world data while incorporating differential privacy mechanisms to protect sensitive information. Its significance lies in providing high-quality synthetic data for research and analysis without compromising patient privacy, making it particularly valuable in the healthcare sector.

Overall, these scholarly contributions offer new insights, methodologies, and tools that address the pressing challenges in privacy-preserving data management. They remind us of the shared responsibility among researchers, practitioners, and policymakers to guide data management towards outcomes that are ethically sound and universally beneficial.

We extend our gratitude to all authors for their valuable contributions to this special issue. Their work not only advances our understanding but also paves the way for future innovations in the realm of secure and private data management.

Haixun Wang
Instacart

Letter from the Special Issue Editor

Privacy-preserving data management is a critical and timely topic in our increasingly data-driven world. As organizations across various sectors, ranging from healthcare and finance to social media and government, collect and process vast amounts of data, the need to protect individual privacy has never been more crucial. Privacy-preserving data management encompasses a range of techniques and practices designed to ensure that sensitive information remains secure while still allowing data to be used for beneficial purposes such as research and analytics.

The importance of privacy-preserving data management cannot be overstated. On one hand, data drives innovation and growth, enabling breakthroughs in medical research, enhancing customer experiences, and optimizing operations across industries. On the other hand, mishandling or exposing sensitive data can lead to significant harm, including identity theft, financial loss, and erosion of trust. Regulatory frameworks such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA) highlight the need for stringent data protection measures, but compliance alone is not enough. Effective privacy-preserving data management requires a proactive approach, integrating advanced technical solutions with sound governance practices.

This special issue features a collection of six papers from expert researchers that push the boundaries of our current understanding and capabilities, reflecting diverse perspectives on privacy protection in the management of data. We start with a paper by Sohn *et al.* that surveys the latest advancements in secure and private database systems. With the surge in data collection and cloud computing, the authors highlight critical privacy challenges and solutions. They delve into technologies like differential privacy, secure multiparty computation, and zero-knowledge proofs, explaining how these methods protect sensitive data while allowing for meaningful analytics. The paper serves as a practical guide for navigating the complexities of implementing privacy-preserving techniques in database systems. The second article, by He and Zhang, explores the application of differential privacy to provenance data, which are records that describe the history of data, including how it was collected, processed, and used. The authors provide a comprehensive framework that ensures privacy while preserving the utility of the data for various applications. The authors present detailed methodologies and case studies that highlight the effectiveness of their approach in real-world scenarios. Next, we feature a paper by Gavidia-Calderon *et al.* that introduces SQLSynthGen, a method for generating synthetic relational datasets, particularly focusing on healthcare data from National Health Service (NHS) hospitals in the UK. The tool addresses the dual need for data fidelity and privacy, providing a white-box approach to synthetic data generation that includes differential privacy mechanisms for enhanced security. The fourth paper, by Mao *et al.*, provides a comprehensive survey of differential privacy applied to time series data. The paper discusses the unique challenges of protecting privacy in time series due to their volume and temporal correlations. The survey covers various techniques to balance privacy and utility, and it highlights the open challenges and future directions in this evolving field. The fifth contribution, by Monir *et al.*, reviews adaptive techniques in Differentially-Private Stochastic Gradient Descent (DP-SGD), focusing on improving the privacy-accuracy trade-off. It also discusses the data management challenges associated with the deployment of DP-SGD, providing insights into enhancing computational and memory efficiency. Finally, the last article by Islam *et al.* focuses on text data, and examines the intersection of bias, fairness, and differential privacy in NLP models. The article provides an in-depth analysis of how differential privacy impacts the fairness of NLP models and offers strategies to mitigate potential biases, ensuring that privacy-preserving techniques do not inadvertently harm model performance or fairness.

Overall, these works offer new insights, methodologies, and tools that address the pressing challenges that we face in the filed of privacy-preserving data management. We would like to thank all authors for their valuable contributions.

Xiaokui Xiao
National University of Singapore.

Everything You Always Wanted to Know About Secure and Private Database Systems (but were Afraid to Ask)

Donghyun Sohn, Xiling Li, Jennie Rogers
{donghyun.sohn,xiling.li,jennie}@northwestern.edu

Abstract

Individuals and organizations are accumulating data at an unprecedented rate owing to the advent of inexpensive cloud computing. Data owners are increasingly turning to secure and privacy-preserving collaborative analytics to maximize the value of their records. In this paper, we will survey the state-of-the-art of this growing area. We will describe how researchers are bringing security and privacy-enhancing technologies, such as differential privacy, secure multiparty computation, and zero-knowledge proofs, into the query lifecycle. We also touch upon some of the challenges and opportunities associated with deploying these technologies in the field.

1 Introduction

With the rise of inexpensive cloud computing and its highly available storage, we are collecting data at an unprecedented rate. Businesses, governments, and other organizations are increasingly outsourcing their data management operations accordingly. They are also realizing new opportunities for analytics in important domains such as clinical research, public policy, and more. There are, however, (at least) two issues that prevent this burgeoning field from reaching its full potential. First, data owners are concerned about the security and privacy of outsourcing their private records, especially regarding their liability and compliance obligations. Second, despite the ubiquity of data, records remain fractured among multiple independent databases. Without putting data in context, these systems may produce query answers that are incomplete and misleading.

To address these challenges, the database community has been very actively researching techniques that protect confidential records in a relational database by architecting security and privacy (S&P) techniques as first-class citizens in their operations. This is happening at every step of the query lifecycle, as shown in Figure 1. In this paper, we will look at systems that are secure; they protect their records from unauthorized access. We will also describe ones that are privacy-preserving–they release information about a dataset while protecting their individual records from reconstruction attacks. We will also delve into outsourced verifiable systems, where an untrusted party provides authenticated query answers over private data. These problems are partially overlapping and we refer to solutions in this space as trustworthy database systems. This myriad of S&P options for databases might seem bewildering to newcomers. In this paper, we will offer a field guide to these emerging systems. We will describe their guarantees and outline the advantages and disadvantages to competing approaches.

We organize the rest of this paper as follows. We review the fundamentals of trustworthy database systems in Section 2. After that, we will survey the state-of-the-art for secure query processing in Section 3. We will

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

then progress on to examine techniques for integrating differential privacy into the query processing pipeline in Section 4. After that we will describe techniques for verifiable query processing in Section 5. Last, we discuss how researchers are assembling these building blocks into privacy-preserving database operators in Section 6.

2 Background

In this section, we review the fundamental concepts that underpin trustworthy database systems. These systems are built upon several techniques from the security and privacy community. They range from cryptographic primitives for protecting data at rest and during query evaluation to frameworks that quantify and control the information leakage associated with running a query. This survey will explore database systems that protect their records' privacy or integrity from one or more adversaries. Systems that center on protecting the privacy of user queries, as opposed to private data, are beyond the scope of this paper. Examples of this include private information retrieval [25, 92], searchable symmetric encryption [51], and private function evaluation [124].

Symbol	Meaning
$\mathcal{Q}$	A query submitted to a trustworthy DBMS
$\mathcal{D}$	A database over which we evaluate $\mathcal{Q}$
$\mathcal{M}$	The mechanism that computes $\mathcal{Q}$
$\mathcal{R}$	The result of $\mathcal{Q}$ computed over $\mathcal{D}$
$\mathcal{C}_{ver}$	A checker for verifying $\mathcal{R}$

Table 1: Notation for query workflow

Early privacy-preserving database research anonymizes private data [65, 89, 78, 66]. For example, k-anonymity [115] only releases a record, or aggregate thereof, if there are at least $k-1$ others that are indistinguishable from it. These techniques give the misleading intuition that individuals "hide in the crowd" in an anonymized data release, but research indicates that the widespread availability of auxiliary data and reconstruction attacks makes this position unsustainable [86, 91, 106]. We touch upon anonymization here because, at the time of this writing, this approach enjoys regular use in numerous domains including US healthcare [90] and education [24, 111] data protection and we expect to see this continue in the near-term as more effective approaches mature and gain traction. We will focus on these next-generation techniques in the remaining text.

Notation. We outline the notation we will use in this text in Table 1. Our query workflow begins with a client submitting their query, $\mathcal{Q}$, to a trustworthy DBMS. They wish to run this SQL statement against a private database, $\mathcal{D}$. If they are querying n private engines, they query $(\mathcal{D}_1, \ldots, \mathcal{D}_n)$. The client may or may not be trusted with the records of $\mathcal{D}$, as we will cover shortly. The database evaluates the query with a mechanism, $\mathcal{M}$, that produces its S&P-preserving result, $\mathcal{R}$:= $\mathcal{M}$ $(\mathcal{D})$. If we are accompanying $\mathcal{R}$ with a proof of its authenticity, then we compute a boolean function, $\mathcal{C}_{ver}(\mathcal{D}) \in \{0, 1\}$, and it returns 1 to the client if it verifies $\mathcal{R}$ successfully.

2.1 Query Lifecycle

Figure 1 shows the broad steps with which a database system evaluate a query and where they integrate assorted privacy and security-enhancing techniques into this workflow. This figure is inspired by [3]. The dotted lines denote steps in the process that may leak information about a database's private records. We expect the initial input databases from one or more data providers or data owners are correct and complete at setup time. A clients queries the records of one or more private databases. The party computing the query answer–this could be the data owner themselves, an untrusted cloud service provider used for

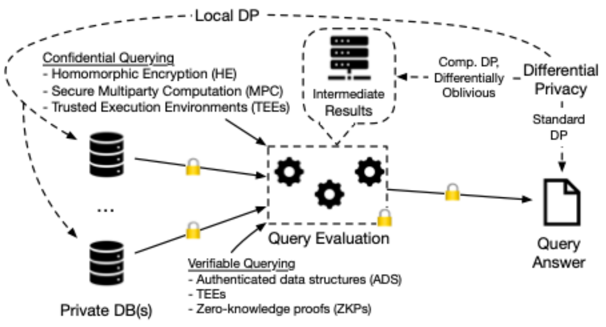
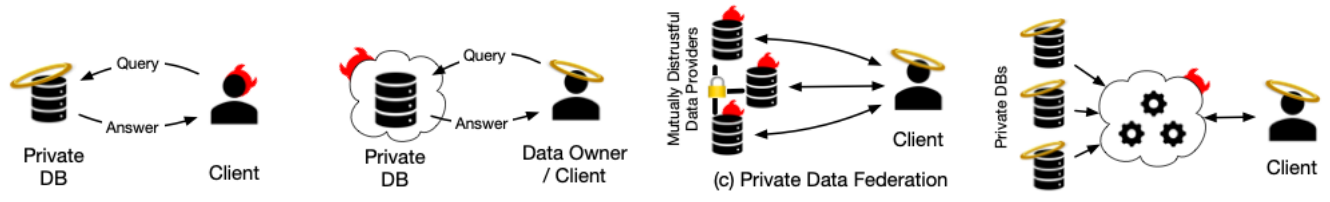

Local DP
Confidential Querying
- Homomorphic Encryption (HE)
- Secure Multiparty Computation (MPC)
- Trusted Execution Environments (TEEs)
Intermediate Results
Comp. DP, Differentially Oblivious
Differential Privacy
Standard DP
...
Query Evaluation
Query Answer
Verifiable Querying
- Authenticated data structures (ADS)
- TEEs
- Zero-knowledge proofs (ZKPs)
Private DB(s)

Figure 1: Trustworthy DBs in the query lifecycle



((a)) Client-Server ((b)) Outsourced Storage ((c)) Private Data Federation ((d)) Outsourced Computation

Figure 2: Reference architectures for trustworthy database systems.

outsourcing or others–may have access to this information leakage. We delve into specific trustworthy database architectures and the S&P-preserving techniques that power them in the upcoming sections.

2.2 Trustworthy DBMS Architectures and Roadmap

Throughout this text, we will reference several common architectures for query evaluation. In this section, we will describe trustworthy database systems in terms of the four architectures shown in Figure 2. Our goal is to provide a guide for future systems builders in selecting the most suitable one for their setting. Admittedly, these frameworks are partially overlapping. We break them up as shown for ease of exposition.

A client-server architecture simply enables the client to send their $\mathcal{Q}$ to the server with a private database and receive its answer, $\mathcal{R}$ as in Figure 2(a). For example, if the client is untrusted and the data provider is trusted, then the latter might protect their query answers from reconstruction by returning noisy versions of the true query answer using differential privacy. We will cover this approach in Section 4.

The outsourced storage architecture, depicted in Figure 2(b), starts with one or more untrusted cloud servers that have ample storage and limited compute resources. A data owner or trusted client outsources their storage operations to this platform to keep their data confidential. These systems support a key-value store-esque interface. This is challenging because even when the database is stored encrypted, side-channel information such as memory access patterns can reveal some or all of the DB's contents [48, 58, 80]. We introduce oblivious RAM in Section 3.2; it is the main starting point for systems in this space. If the client seeks only verifiable results, and they have access to more compute power on the server side, they might engage in verifiable querying as described in Section 5.

The private data federation (PDF), shown in Figure 2(c), enables two or more mutually distrustful parties to compute $\mathcal{R}$ over the union of their private records without disclosing their private records to anyone. This secure query evaluation may take place within cryptographic protocols evaluated among the data providers, described in Section 3.1 or in trusted hardware, as detailed in Section 3.4.

The outsourced computation setting in Figure 2(d) is when one or more data providers outsource their storage to the untrusted cloud and they delegate all query evaluation to this platform. Here, the client and data providers are both trusted. We discuss approaches to solving this challenge in Section 3. Similar to the PDF, platforms can securely compute in hardware or software. This setup introduces another option: homomorphic encryption, or computing over encrypted data, as described in Section 3.3.

3 Secure Query Processing

In this section, we will examine techniques for query processing on outsourced databases, Figure 2(d), and private data federations, Figure 2(c). We group together these two architectures because there is strong overlap in their approaches and we highlight their differences we go along. We start with the strongest guarantee, oblivious querying, and then introduce popular relaxations to this.

3.1 Oblivious Querying

A program is oblivious if its data access patterns and instruction traces are independent of their private input data. Oblivious algorithms prevent an attacker from inferring information about a database by observing its queries as they are evaluated. An obliviously-executed query divulges nothing about its private inputs, except that which can be deduced from its results.

Researchers prove that a program is oblivious using the "real world, ideal world" paradigm [21]. Say that we are computing query $\mathcal{Q}$ with mechanism $\mathcal{M}$ over database $\mathcal{D}$ and there exists a probabilistic polynomial time simulator, Sim that receives $\mathcal{D}$', an arbitrary database that is not $\mathcal{D}$ but shares its schema and table cardinalities. The observable traces of $\mathcal{M}$ should be computationally indistinguishable from those of Sim. In other words, $Traces(\mathcal{M}, \mathcal{D}) \equiv Sim(\mathcal{M}, \mathcal{D}')$. This paradigm makes it possible to compose independently developed building blocks, such as the oblivious query operators, into a query plan with strong end-to-end guarantees.

There are a few common mechanics in oblivious database operators [10, 138, 119, 73, 4]. To keep their instruction traces oblivious to their private inputs, these programs evaluate both branches of private if-then statements, retaining only the one indicated by its secret conditional. Similarly, they do not admit early termination of loops. They conceal the selectivity of database operators with dummy tuples that replace rows that would be filtered out in an operator so that a curious observer cannot deduce anything about a function's input data. These engines typically represent this feature with a dummy tag or bit appended to each row denoting whether it should be included in any subsequent calculations. For example, an oblivious filter visits every row in a table and applies its private predicate. If the row satisfies the selection criteria, the oblivious if-then will update the row's dummy tag. Otherwise, it will simply overwrite the dummy tag with its previous value to remain oblivious. We will describe oblivious database operators in detail in Section 6.

Oblivious querying incurs a substantial performance penalty because its operators hide their data access patterns and any data-dependent changes in their control flow. With no additional info, queries with cascading joins must pad their output to the maximum possible size (the cross-product of their inputs) to conceal their selectivity. Cumulatively, leads to a blow-up in their cardinalities increasing the cost of any subsequent computation. As such, many oblivious database operators engage in selective information disclosure such as revealing the true cardinality of joins [63, 138] by default while offering full-padding if desired. If this is the root (last) operator in a query tree and the parties will receive $\mathcal{R}$, then this is a sensible trade-off. This picture gets more complicated if it is an intermediate result.

Secure multiparty computation (MPC) [44, 74] enables two or mutually distrustful parties to jointly compute over their private inputs obliviously. Although early applied MPC protocols often implemented a special-purpose function (such as linear regression) to tackle the breathtaking overhead associated with general-purpose protocols, most modern systems compile their program logic into circuits [53]. These garbled circuits reduce $\mathcal{Q}$ into a series of gates, e.g., AND and XOR gates. The protocol evaluates the circuit obliviously by traversing it in topological order. They provide a Turing complete springboard for evaluating ad-hoc queries. Some protocols use arithmetic gates instead of logic ones. Performing all private computation within garbled circuits makes it possible to seamlessly compose the security guarantees of multiple, independently-developed components into a single proof under universal composability [21].

Private Data Federations. There is a growing need to analyze information from multiple sources through a unified querying interface while addressing privacy concerns and complying with numerous privacy laws. A PDF, as in Figure 2(c), integrates multiple autonomous database systems owned by mutually distrustful parties to query them as if they were a single query engine. PDFs offer a shared schema that has table definitions agreed upon by all data providers at setup time. It specifies the security level needed for each column. Many PDF frameworks evaluate their queries under MPC. This ensures that no unauthorized data is disclosed to other data providers participating in a secure query, while also minimizing the operations that $\mathcal{M}$ must perform under MPC by pushing them down for local evaluation [10, 119, 73].

Data providers store private data in their own databases and compute query outputs in a privacy-preserving manner. In PDFs, the process operates as follows: the client translates $\mathcal{Q}$ into the corresponding cryptographic protocols with which they will execute it. They then send these instructions to all participating data providers. The data providers then locally compute any public query operators over their private records. They then encode their selected private input rows for the operators that require secure, distributed evaluation over their unioned intermediate results. They execute their oblivious operators by passing encrypted messages among themselves. The data providers then send their share of $\mathcal{R}$ to the client over an encrypted link. After receiving shares from all of the computing parties, the client assembles $\mathcal{R}$.

There are numerous threat models for MPC protocols, and they are detailed in [74]. A semi-honest party adheres to the protocols, but may attempt to learn additional, unauthorized information from observing the MPC protocol. Semi-honest database systems include SMCQL [10], Conclave [119], and Hu-Fu [116]. On the other hand, a malicious party both seeks to reveal information about a query's private inputs and they may deviate from the MPC protocol arbitrarily. Senate [96] implements a PDF in the maliciously secure setting. Some engines are starting to support multiple protocols or mixed models by compiling queries into abstract circuits (or methods) and applying different protocols for different settings. SCQL [4] and VaultDB [107, 112] take this approach. Naturally, protocols with stronger guarantees incur more overhead for query evaluation, and this design choice depends on the setting in which they run.

Outsourced Computation. MPC facilitates querying over the unioned data of 2+ independent private DBs. We need one more step to extend this technology to the outsourced computation setting in Figure 2(d). To distribute trust over multiple outsourced hosts, a client may secret share their private inputs and send shares of them to all computing parties. Here no party can reconstruct the secret unless the party colludes with a subset of parties. Before starting to evaluate $\mathcal{Q}$'s garbled circuits, the parties participate in an oblivious transfer protocol to encode their data as wire labels for its logic gates. This process is analogous to public-key cryptography where at the end of it each party holds an encrypted copy of the database without access to the key with which to decrypt it and none has access to the plaintext data except its initial owner.

Platforms in the outsourced computation model have also been adopting MPC protocols. The earliest work (to the best of our knowledge) in this space computed aggregates semi-honestly under 3-party computation [17]. SDB [128, 55] created a hybrid model where the private data is secret-shared among the data owner and the cloud service provider. For the semi-honest setting, Secrecy [73] supports 3-party secure computation over ad-hoc SQL queries. RESCU-SQL [69] uses maliciously secure, n-party MPC protocols to protect outsourced data in the zero-trust cloud.

3.2 Oblivious RAM for Outsourced Storage

We now turn our attention to the outsourced storage setting shown in Figure 2(b) in Section 2.2. Here, the client wishes to outsource the of their private database to the untrusted cloud. If they simply encrypted their data and issued read and write requests against the specific records as they access them, then this will expose their data access patterns and make their data vulnerable to side-channel leakage attacks [58, 87, 48]. Unless otherwise specified, these systems have a key-value store-style API. Oblivious RAM [43] transforms a client's read and write requests into ones that are independent of their true memory access patterns. When a client requests a block, b_i, from their database, the ORAM rewrites it as a series of reads and writes that 1) shuffle their storage, and 2) pad their request with dummies to conceal the position of their request in the database. This makes the distribution of their I/O requests oblivious to their true memory access patterns. It also prevents an adversary from deducing the frequency of accesses to a specific b_i.

Early work in outsourced storage centered on ORAM constructions themselves, with tree-based ones [114, 103] emerging as the dominant paradigm in practice. Several systems have generalized this work to distributed ORAMs, including Shroud [75], ObliviStore [113], and TaoStore [110]. Snoopy [30] integrates trusted hardware into their design to parallelize oblivious reads and writes. Whereas ORAM makes all access requests indistinguishable from one another, frequency smoothing relaxes this requirement to make the frequency with which individual b_is are requested uniform. Waffle [79] and Pancake [47] are examples of this approach. They do so by replicating popular objects and padding their I/O requests with dummies.

The systems described above are all linearizable; they reason about concurrency at the granularity of one object at a time. Additional mechanisms are needed for ACID transactions. Obladi [29] is an ORAM-backed storage engine that parallelizes ACID transactions on untrusted ORAM servers. Treaty [42] is a natural extension of this work obviates the need for a trusted proxy with trusted hardware.

3.3 Homomorphic Encryption for Outsourced Computation

Homomorphic encryption (HE) [41] enables a system to compute over encrypted data without decrypting it. HE and MPC are close cryptographic cousins. Rather than distributing trust over multiple parties, HE enables data owners to outsource their operations to one host with a variation of the outsourced computation architecture in Figure 2. Here, the data provider encrypts their records and uploads them to the untrusted cloud (without providing the decryption key) and the client issues queries against the encrypted databases. Whereas MPC enables parties to pipeline their circuit evaluation, i.e., compute each gate and discard intermediate wire labels when they are no longer needed, HE protocols evaluate a materialized circuit to produce $\mathcal{R}$ and send it to the client. This is more efficient for straightforward operators such as filter, but may reveal scalability challenges for ones with deeper circuits such as aggregating over a large group of tuples. Fully homomorphic encryption (FHE) support circuits, and thus ad-hoc queries, without leaking information about the encrypted data. FHE has a very high performance cost, typically several orders of magnitude slower than running in plaintext. These schemes have seen limited adoption in databases in practice with the only known implementation for databases targeting aggregation alone [104]. HE has several relaxations to bridge this performance gap. Some partially homomorphic encryption (PHE) schemes offer better performance in exchange for reduced security guarantees, such as order preserving encryption [18, 1] and deterministic encryption [19, 14]. Others support reduced expressiveness with higher performance, such as additive HE [93] and multiplicative HE [38, 105].

Database researchers have invested substantial research effort into bringing these encryption schemes to outsourced databases with a variation of the architecture in Figure 2(d). The data owner encrypts $\mathcal{D}$ using HE and uploads them to the untrusted cloud server. The client submits $\mathcal{Q}$ to the server and it computes $\mathcal{R}$ over its encrypted copy of $\mathcal{D}$. [1] introduced order-preserving encryption for answering database queries. CryptDB [97] and Monomi [117] targeted HE for outsourced databases for OLTP and OLAP workloads, respectively. The latter identified lightweight mechanisms for moving some of the query evaluation client-side to circumvent the performance limitations of FHE. Unfortunately, these PHE schemes have substantial side-channel leakage associated with executing queries over them [87, 15]. This is similar to the issues we described for non-oblivious access to outsourced storage above. SEAL [32] tackles some of this leakage with adjustable leakage that hides $\binom{n}{k}$ bits out of a search key by introducing a generalization of ORAM.

3.4 Trusted Execution Environments for Outsourced and Federated Querying

Trusted Execution Environments, or TEEs, create an isolated computing platform within an untrusted computer using trusted hardware. They have encrypted private memory with which they runs sealed code that is confirmed to be only the code given by the client or a proxy thereof (such as a trustworthy DBMS). Hence, code run within TEEs are verifiable by virtue of running within a secure enclave, such as Intel SGX [33] and ARM TrustZone [95]. Clients delegate their program executions to trusted hardware, safeguarding their private data without the need for encryption. The main advantage is its efficiency, as it avoids the significant computational and communication overhead typically incurred by cryptographic primitives, making it more practical for real-world scenarios. Secure enclaves alone are not a panacea. For example, instruction traces leak memory access patterns, allowing eavesdroppers to infer private data from them [58, 80].

TEEs are seeing increased use for evaluating queries over confidential data the cloud or outsourced computation in Figure 2(d). First-generation systems relied on software-hardware co-design to build TEEs with bespoke algorithms for query evaluation such as TrustedDB [8] from IBM and Microsoft's Cipherbase [5]. As SGX and other enclaves became widely available and cloud-ready, researchers created secure enclave extensions to well-known DBMSs. For example, StealthDB [118] builds atop PostgreSQL and EnclaveDB [98] runs within Hekaton. In addition, since the resident host running the enclave can observe its memory access patterns, researchers have been building oblivious algorithms for use inside the TEE such as ObliDB [39] and Opaque [138]. This approach has also been appearing in TEE-based PDFs such as [31].

4 Differential Privacy

If the client is untrusted, the data provider may call for guarantees that prevent them from using query answers to infer information about their private input records. Speaking informally, Differential Privacy (DP) [34, 35] protects private data from construction attacks by injecting carefully calibrated noise into their query answers to control their resulting information leakage before returning them to the client. An algorithm satisfies differential privacy if its output is approximately the same over a database, $\mathcal{D}$, as it would be over a neighboring database differing by one record, $\mathcal{D}$'. This workflow, of a data provider or trusted curator noising query answers before releasing them to the client, is known as Standard DP or SDP [121]. More broadly, if a secure database reveals precise, un-noised query answers, this creates unbounded information leakage [60]. SDP works on the client-server model in Figure 2(a).

Before answering their first query over $\mathcal{D}$, the data owner selects a privacy budget, ϵ, with which they limit the information they leak in aggregate to the client or clients by answering their queries. Recall that $\mathcal{R} := \mathcal{M}(\mathcal{D})$. In its simplest form, if the client queries a private database n times with mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_n$, and $\mathcal{M}_i$ has ϵ_i, their net privacy loss is $\sum_{i=1}^{n} \epsilon_i$. Researchers have proposed several frameworks for designing efficient DP mechanisms [56, 133, 126]. Also, the database community has invested significant research effort into automatically deriving SDP answers to SQL queries with robust e2e privacy guarantees [109, 81, 99, 57, 62, 61].

Choosing ϵ for a given database and query workload reveals a trade-off between utility and privacy. A larger ϵ means that $\mathcal{M}$ will inject less noise into $\mathcal{R}$. In other words, more accurate query answers result in a greater privacy loss for $\mathcal{D}$. Selecting an ϵ with sufficient data protection while producing useful query answers is a challenging topic [64] and the subject of ongoing research [36, 72, 88]. A conservative heuristic is to restrict ϵ to values less than or equal to one [129].

4.1 Computational Differential Privacy

One major shortcoming of SDP is that assumes there is a single trusted data curator noising $\mathcal{R}$. Computational DP (CDP) [13, 83] relaxes SDP's strong information-theoretic guarantees to a weaker, probabilistic polynomial time adversary in exchange for less noisy results and support for distributed data. CDP quantifies information leakage when two or more parties are computing a function over their encrypted and unioned private inputs, $\mathcal{D}_*$. It frames information leakage about $\mathcal{M}$'s intermediate results as a differentially private view of $\mathcal{D}_*$ and assigns some share of the privacy budget to it. For example, if we are computing a CDP filter $\mathcal{R} := \sigma(\mathcal{D})$ we may start by running the oblivious evaluation described above. We then generate a noisy version of its true output cardinality within the oblivious algorithm for release, $|\widetilde{\mathcal{R}}|$ and obliviously delete dummy rows. This will make any parent operators run faster because they compute over fewer dummies.

CDP introduces a third dimension to our DP trade-off: performance. If we allocate more privacy to computation, the client may get their query answer faster but with reduced accuracy because we spend some privacy releasing information about $\mathcal{M}$'s intermediate results, i.e., its noisy intermediate cardinality. There's been significant research interest in using CDP to accelerate privacy-preserving query evaluation in PDFs [11, 12, 54].

CDP extends to the outsourced storage setting. Allowing an untrusted cloud service provider to view noisy versions of the client's memory access patterns enables this trade-off on efficiency and privacy. DP-ORAM [120] relaxes ORAM's full-oblivious guarantees to CDP ones. ϵpsolute [59, 16] parallelizes CDP I/O requests among multiple ORAMs. CDP has also been used to speed up outsourced computation for analytics [26] and for querying growing databases [134, 123, 122] in the untrusted cloud.

4.2 Differential Obliviousness

Another DP relaxation gaining traction in the database community is differential obliviousness [22, 27, 100, 101] (DO). A DO algorithm offers a DP view of its memory traces by leaking approximate versions thereof. This is related to CDP, with the latter approximating discrete views of $\mathcal{D}$. To continue with our filter running example, a simplified version of the DO filter in [100] partitions $\mathcal{D}$ into batches of length B where B is polylog of $|\mathcal{D}|$. In lieu of running an oblivious filter, it writes to the output buffer of $\mathcal{R}$ one batch at a time maintaining a cursor for these writes. For each block b_i it computes a DP approximation of its prefix sum (how many rows are selected), this provides a range of the count of the rows b_i will emit to $\mathcal{R}$. It then writes to B slots in the output buffer with a mix of real and dummy rows and advances the cursor to the position indicated by the lower bound of its DP prefix sum.

DO strikes a balance between full-oblivious ones and ones with unbounded information leakage. It admits secure algorithms with performance properties comparable to streaming or pipelined operators and it boasts better cache efficiency than approaches that materialize their entire $\mathcal{R}$ before noising it. On the other hand, DO mechanisms are non-trivial compose [139], and they need to maintain the notion of neighbor-preserving differentially oblivious datasets between an operator's input and output relations to compose a DAG of them. Addressing this challenge is a topic of ongoing research.

4.3 Local Differential Privacy

One more approach to limiting information leakage from querying private data is injecting noise into the private data before querying it. Local DP [130] (LDP) removes the need for a trusted data curator by noising it one row at a time. There has been considerable research in incorporating this into database operations [28, 49, 108, 125, 131]. This is different from DP synthetic data generation [20, 40, 77], which creates a new dataset with values that mimic the distribution of a private one.

LDP is attractive for aggregating "wisdom of the crowd" statistics, such as collecting new words and phrases for auto-complete as they enter the popular lexicon and identifying software bugs from noisy bug reports. It also frees data collectors from the liability of storing raw user data, which may garner subpoenas or attempts to steal it. Also, since the data are noised upfront, clients may query it as many times as they wish without eroding the privacy budget. On the other hand, because the data are perturbed one row at a time, the algorithm needs to add $O(\sqrt{n})$ to each entry for n individuals contributing to $\mathcal{D}$, whereas SDP would simply noise the true count independent of the participant count [121].

5 Verifiable Querying

Data owners are increasingly outsourcing their data management to the untrusted cloud. With verifiable computing when an untrusted server answers a client query $\mathcal{Q}$ with $\mathcal{R}$ over a private database $\mathcal{D}$, they participate in a protocol to convince the client (with high probability) that $\mathcal{R}$ is correct and complete. Here we have two participants: a prover $\mathcal{P}$, the cloud service provider, and a verifier $\mathcal{V}$, the client. If $\mathcal{C}_{ver}(\mathcal{D})$ is a function with which $\mathcal{P}$ and $\mathcal{V}$ authenticates $\mathcal{R}$ with the following guarantees:

- Completeness. If $\mathcal{P}$ faithfully executes $\mathcal{Q}$ then $\mathcal{C}_{ver}(\mathcal{R}) = 1$. $\mathcal{P}$ accepts honest proofs.
- Soundness. If $\mathcal{P}$ outputs an incorrect $\mathcal{R}$, $Pr[\mathcal{C}_{ver}(\mathcal{R}) = 1] \leq \epsilon$, where ϵ is a negligible probability.

Systems such as IntegriDB [137], Concerto [6], CorrectDB [9], VeriDB [140] and vSQL [135] use VC to guarantee data integrity for querying.

There are several approaches to creating verifiable query answers. With an Authenticated Data Structure, $\mathcal{P}$ generates a proof that accompanies $\mathcal{R}$ to validate that it is complete and sound. The two main methods that instantiate ADS are tree-based and signature-based methods. The tree-based method builds a binary tree for a database, where each leaf node stores a digest about tuples in the database, and its internal nodes are digests that summarize its children. When evaluating a query, the database sends the query answer along with a set of digests for the relevant nodes to the client, who has the digest of the root node, to verify the answer by reconstructing the path to match the root node. IntegriDB is such a system that applies a dynamic tree-based ADS tailored for a specific set of queries, allowing the client to query and verify $\mathcal{R}$ using precomputed digests aided by the accompanying proof. On the other hand, signature-based methods constructs a set of signatures for tuples in the database. During query evaluation, the cloud-hosted database sequentially collects a set of signed tuples that it accesses. Then the client verifies each tuple that no unwanted tuple is included and no necessary tuple is missed.

Similarly, Concerto, CorrectDB and VeriDB utilize ADS-based verifiable computing to verify their queries, but they compose ADSs with TEEs benefit from greater efficiency than working solely with cryptographic primitives (see Section 3.4). Concerto is a concurrent key-value store, while VeriDB and CorrectDB support ad-hoc SQL queries for relational databases.

With interactive proofs [137], $\mathcal{P}$ and $\mathcal{V}$ work together to confirm the authenticity of a statement $C(w)$ over a witness w via multiple rounds of challenge-response messages, thereby incrementally ensuring correctness and soundness. This VC approach is an alternative to ADS. Similar to MPC, these proofs work in the circuit paradigm–they express their logic as gates. In our context, w is the private database $\mathcal{D}$. $\mathcal{P}$ and $\mathcal{V}$ interactively establish w as a commitment of $\mathcal{D}$. This forms the immutable starting point for $\mathcal{P}$'s proof for $\mathcal{Q}$. Upon receiving the last respone $\mathcal{R}$ from $\mathcal{P}$, $\mathcal{V}$ accepts it iif $\mathcal{C}_{ver}(\mathcal{R}) = 1$. In other words, $\mathcal{V}$ rejects immediately if it receives any unconvincing response from $\mathcal{P}$ during the interaction. In contrast to ADS-based systems mentioned above, vSQL is more expressive by supporting a wider range of SQL queries through IPs. Although interactive proofs incur significant communication overhead between the client and server, vSQL operates with nearly the same efficiency as those systems.

However, the client might attempt to obtain extra information that it is not authorized to access. For example, VC-backed systems with ADS and interactive proofs ensure the integrity of an outsourced database, but they reveal the data owner's records to the cloud service provider. Moreover, TEE-based systems may leak memory access patterns through program traces, as discussed in Section 3.4. Therefore, the aforementioned systems are inadequate to protect private data on an untrusted server. For this we need to add one more guarantee to our stack, zero knowledge [45, 46]:

- Zero knowledge. If $\mathcal{C}_{ver}(\mathcal{R}) = 1$, $\mathcal{V}$ learns nothing other than the fact $\mathcal{R}$ was computed correctly.

To provide such a stronger guarantee in verifiable querying, the database community has adopted *zero-knowledge proofs* for SQL queries [136, 70] to offer privacy and confidentiality for query answering. This guarantee is similar to the one we saw for obliviousness: $\mathcal{V}$ learns nothing more than $\mathcal{R}$ and that which can be deduced from it. The main difference here is that $\mathcal{P}$ is able to prove properties of intermediate query results, such as proving a sorted list of tuples is monotonically increasing, rather than evaluating the entire program logic in circuits. Overall, ZK proofs enable $\mathcal{P}$ to convince $\mathcal{V}$ of the query answer $\mathcal{R}$ by authenticating it with a proof, without revealing any additional information beyond the information that could be derived from $\mathcal{R}$. This means we could simultaneously address both data integrity and data privacy for the outsourced database.

There are two main flavors of zero-knowledge proofs: *interactive* and *non-interactive* ZK proofs. Interactive ones are analogous to MPC, where $\mathcal{P}$ and $\mathcal{V}$ verify a circuit one gate at a time using pipelined gates. In contrast to IPs, ZK proofs divulges no additional information about w beside the truth of $C(w)$ to $\mathcal{V}$ while IPs do not hide w from $\mathcal{V}$.

Zero-knowledge extension of vSQL [136] and ZKSQL [70] utilize interactive ZK proofs. While the former remains theoretical as a pioneering effort, ZKSQL optimizes computationally expensive operators for practical use, such as sort and join, reducing their respective complexities from $O(n \log n)$ and $O(n^2)$ to $O(n)$. This optimization is achieved through $\mathcal{P}$'s local computation and interactive verification with ZK set operations, where the set-based proofs are represented by polynomials instead of circuits, unlike conventional interactive ZK proofs. For example, in sorting, we can only check if the result contains the same tuples as the unsorted table using ZK set equality when the result confirms to be monotonically increasing or decreasing.

Conversely, non-interactive ones construct a monolithic circuit for their query in a single step, providing a single proof that $\mathcal{P}$ sends with $\mathcal{R}$. They require no additional interaction between $\mathcal{P}$ and $\mathcal{V}$ and are widely used in blockchain applications. However, a significant drawback is the large proof size, which can be memory-intensive due to the one-shot construction. Despite potentially being more efficient than the interactive ones due to reduced communication overhead, we do not recommend this approach for systems aiming to scalability.

In addition to the single prover-verifier system, there are also distributed or decentralized verification systems that build atop blockchains [37, 2, 94, 132]. They guarantee auditability, accountability, and traceability on the ledger during data sharing across mutually distrustful parties, but their combined records are largely accessible to all participants on the chain thus they are beyond the scope of this paper.

6 Privacy-Preserving Database Operators

This section introduces privacy-preserving database operators. They are crucial for secure and privacy-preserving query evaluation.

We start with oblivious operators. Recall from Section 3.1 that they guarantee that their memory access patterns and program traces do not disclose any information about their private inputs. These operators ensure that even if an adversary knows $\mathcal{Q}$ and observes its execution under $\mathcal{M}$, they learn no additional information from participating in its evaluation. [7] introduced the earliest formal definitions for efficient algorithms for oblivious query processing. This paper covers approaches for selections, joins, and group-by aggregation. It was designed for use in TEEs, although no practical implementation of its results are forthcoming. Since then, there has been a lot of excitement in the research community about creating efficient, oblivious algorithms for database operators [4, 10, 63, 138]. We will also touch upon operators that offer alternative privacy guarantees, such as CDP and DO from Section 4. We will mainly focus on joins in this survey because, in our experience, these tend to be the bottleneck for most secure query workloads. In addition, joins have attracted the most research results to date in terms of operator algorithms proposed.

6.1 Joins

Table 2 presents a comprehensive categorization of privacy-preserving join approaches. This considers several key dimensions: computational complexity, employed methods, type of leakage, number of participating parties, and supporting join types. Notably, the table organizes the joins based on their leakage, establishing distinct categories for oblivious joins, differentially oblivious joins, and differentially private joins.

Oblivious Join. The most strict technique is fully oblivious join, which allows combining data from different sources without revealing sensitive information. Various algorithms have been evaluated in [67] to determine their efficiency and security, as shown in table 2. The study reveals the overall efficiency ranking, wherein PSI emerges as the most efficient, followed by hash approaches. However, the efficiency of NLJ and SMJ can surpass others under high join selectivity. Additionally, optimal join order significantly improves efficiency, highlighting the importance of cost-based query optimization.

DO Joins. To improve efficiency, DO join permits a controlled degree of information leakage about the data while still providing meaningful privacy guarantees based on the relaxed notion of $(\epsilon, \delta)-$differential obliviousness. Key advancements in this area include the DO join [27], Adore [100], Doquet [101], as shown in Table 2. Fully oblivious join algorithms are inefficient due to worst-case padding, resulting in significant performance overheads.

Strategy	Complexity	Compute Method	Leakage	# of Parties	Join Type
Nested Loop	$O(n^2)$	SH-2PC	Oblivious	2: [10, 112]	all
		SH-3PC	Oblivious	3: [119, 73]	
		Mal-MPC	Oblivious	N: [69]	
		SH-2PC	CDP	2: [11, 12]	
		TEE	Oblivious	1: [71], $\geq$ 2: [31]	
NLJ w/semi-join	$(n \times RF)^2$	SH-2PC	Oblivious	2: [10]	equi-join
		SH-3PC	Oblivious	3: [27, 73]	
Index NLJ	$n \log^2 n$	ORAM	Oblivious	N: [23]	all
Sort-Merge Join	$n \log^2 n$	TEE	Oblivious	$\geq$ 1: [63, 39, 138]	equi-join
		SH-3PC	Oblivious	3: [67]	
		TEE	DO	≥ 1 : [27]	
PK-FK SMJ	$n \log n$	TEE	DO	$\geq$ 1: [100]	1:n
Hash Join	n	SH-3PC	Oblivious	3: [84]	equi-join
Hash SMJ	n	SH-3PC	Oblivious	3: [76]	equi-join
PSI join	n	SH-2PC	Oblivious	2: [102, 127];	pk-pk
		Mal-2PC	Oblivious	2: [102]	pk-pk
	$n \log n$	SH-MPC	Oblivious	N: [4]	equi-join
	$n \ln \ln n$	SH-MPC	CDP	N: [85]	equi-join
Partition Join	$n \log^2 n$	TEE	Oblivious	N: [68]	equi-join
	$n \log n$	TEE	DO	$\geq$ 1: [101]	equi-join

Table 2: Comparison of secure join algorithms.

The DO join algorithm addresses this by using $(\epsilon, \delta)-$DO [22], a less strict privacy concept, to achieve efficient joins compared to insecure methods while preserving privacy. This approach provides meaningful privacy guarantees and proves new lower bounds on DO join algorithm performance. Adore employs a similar strategy with a sort-merge join but improves efficiency by using bucket oblivious sort, reducing the sorting complexity from bitonic sort's O($n \log^2(n)$) to O($n \log n$). Additionally, Doquet optimizes join costs through a partitioning approach, significantly improving performance.

CDP Joins. CDP joins provide strong privacy guarantees by ensuring that the inclusion or exclusion of any individual in the dataset trivializes the join operation's outcome. DJoin [85] supports SQL-style join queries across multiple databases using two novel primitives: Blinded, Noised Private Set Intersection Cardinality (BN-PSI-CA) for private intersections with added noise to ensure differential privacy, and Denoise-Combine-Renoise (DCR) which combines noised subset sizes efficiently for privacy-preserving joins on distributed data. This results in an efficient join complexity. Shrinkwrap [11] addresses inefficiencies in PDF by leveraging computation differential privacy to reduce intermediate result padding. It integrates a cost model and privacy budge optimizer to balance privacy and performance. SAQE [12] scales PDF to handle large datasets by combining DP with approximate query processing. It introduces secure sampling algorithm that reduce computation costs and minimize the noise injected into query results.

6.2 Additional Operators

We will now focus on additional operators and frameworks in the trustworthy database system landscape.

Oblivious Aggregate. Oblivious aggregation computes summary statistics over a set of records without revealing how they are partitioned with a `GROUP BY` clause. SAGMA [52] and OBSCURE [50] represent two approaches to this challenge. SAGMA supports secure aggregation grouped by multiple attributes under somewhat holomorphic encryption (SWHE). It allows the user to choose any combination of grouping attributes and privately maps rows to buckets, balancing storage and computational needs. However, SAGMA's main drawback is its need to store an exponentially growing set of monomials because its SWHE encoding supports only one multiplication operation [104]. On the other hand, OBSCURE encodes private rows using order-preserving secret sharing (OP-SS), which maintains data order securely while supporting repeated aggregation queries. OBSCURE also introduces an oblivious result verification mechanism and demonstrates scalability to large datasets, addressing challenges faced by previous secret-sharing or MPC systems. However, OBSCURE's use of OP-SS is not inherently secure and is vulnerable to background knowledge attacks.

Oblivious Filter. QFilter [82] combines an Attribute-Based Access Control (ABAC) model with query processing over secret-shared data. This integration allows for fine-grained access control while preserving privacy. It handles aggregation SQL queries like “count", “sum", and “avg", without requiring inter-server communication, thus securing against honest-but-curious adversaries. The system efficiently handles queries through string matching-based operators, rewriting SQL queries to embed access authorizations and filter unauthorized data during execution. However, QFilter's limitations include its inability to support more complex queries like “min" and “max" functions, “group-by" clauses, or range queries, which limits its applicability in demanding data environments.

SODA. SODA [68] introduces a collection of oblivious algorithms designed for distributed data analytics, including filter, aggregate, and binary equi-join operations. It improves upon previous systems like Opaque [138] by minimizing data padding through a two-level bin-packing strategy. This approach effectively manages input redistribution and handles join product skewness. SODA further avoids expensive global sort primitives by employing low-cost pseudo-random communication to guarantee uniform data traffic. However, SODA does not address issues related to denial-of-service attacks or physical side-channel attacks. Additionally, it does not integrate hardware oblivious memory, which could further protect against side-channel vulnerabilities by hiding access patterns more effectively.

Differentially Oblivious Operators. Adore [100] not only achieves differential obliviousness for joins, but it extends this guarantee to other database operators, like selection and aggregation, by working within secure enclaves. Doquet [101] introduces a framework for DO range and join queries using private data structures. It improves on the efficiency of Adore and oblivious algorithms on SGX. Doquet also addresses access pattern leakage vulnerabilities that were present in Adore, ensuring a more secure implementation than that of its predecessor. Both systems highlight the potential of DO to trade off on privacy and efficiency in query evaluation.

7 Conclusions

In this paper we surveyed the state-of-the art in security and privacy-preserving database systems. We compared the properties of competing frameworks for trustworthy querying, such as secure computation vs trusted hardware for secure querying and authenticated data structures vs interactive proving for verifiable querying. In addition, we described how differential privacy is making its way into nearly all of the steps in the query lifecycle. We highlighted how composing these techniques reveals many subtleties in their S&P guarantees, such as computational differential privacy over secure computation. We closed with an exploration of how these techniques come together to create efficient, privacy-preserving database operators.

References

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pages 563–574, 2004.

[2] L. Allen, P. Antonopoulos, A. Arasu, J. Gehrke, J. Hammer, J. Hunter, R. Kaushik, D. Kossmann, J. Lee, and e. a.

Ravi Ramamurthy. Veritas: Shared verifiable databases and tables in the cloud. In 9th Biennial Conference on Innovative Data Systems Research (CIDR), pages 1–9, 2019.

[3] Ant Group. A comprehensive comparison of various privacy-preserving technologies. `https://www.yuque.com/secret-flow/admin/exgixt72drdvdsy3`. Accessed: 2024-06-12.

[4] Ant Group. Secure collaborative query language. `https://github.com/secretflow/scql`. Accessed: 2024-06-13.

[5] A. Arasu, S. Blanas, K. Eguro, R. Kaushik, D. Kossmann, R. Ramamurthy, and R. Venkatesan. Orthogonal security with cipherbase. In CIDR, 2013.

[6] A. Arasu, K. Eguro, R. Kaushik, D. Kossmann, P. Meng, V. Pandey, and R. Ramamurthy. Concerto: A high concurrency key-value store with integrity. In Proceedings of the 2017 ACM International Conference on Management of Data, pages 251–266, 2017.

[7] A. Arasu and R. Kaushik. Oblivious query processing. ICDT, 2014.

[8] S. Bajaj and R. Sion. Trusteddb: a trusted hardware based database with privacy and data confidentiality. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, pages 205–216, 2011.

[9] S. Bajaj and R. Sion. Correctdb: Sql engine with practical query authentication. Proceedings of the VLDB Endowment, 6(7):529–540, 2013.

[10] J. Bater, G. Elliott, C. Eggen, S. Goel, A. N. Kho, and J. Rogers. Smcql: Secure query processing for private data networks. Proc. VLDB Endow., 10(6):673–684, 2017.

[11] J. Bater, X. He, W. Ehrich, A. Machanavajjhala, and J. Rogers. Shrinkwrap: efficient sql query processing in differentially private data federations. Proceedings of the VLDB Endowment, 12(3), 2018.

[12] J. Bater, Y. Park, X. He, X. Wang, and J. Rogers. SAQE: practical privacy-preserving approximate query processing for data federations. Proceedings of the VLDB Endowment, 13(12):2691–2705, 2020.

[13] A. Beimel, K. Nissim, and E. Omri. Distributed private data analysis: Simultaneously solving how and what. In Advances in Cryptology–CRYPTO 2008: 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings 28, pages 451–468. Springer, 2008.

[14] M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In Advances in Cryptology–CRYPTO 2008: 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings 28, pages 360–378. Springer, 2008.

[15] V. Bindschaedler, P. Grubbs, D. Cash, T. Ristenpart, and V. Shmatikov. The tao of inference in privacy-protected databases. Cryptology ePrint Archive, 2017.

[16] D. Bogatov, G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill. εpsolute: Efficiently querying databases while providing differential privacy. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pages 2262–2276, 2021.

[17] D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A framework for fast privacy-preserving computations. In S. Jajodia and J. Lopez, editors, Proceedings of the 13th European Symposium on Research in Computer Security., volume 5283 of Lecture Notes in Computer Science, pages 192–206. Springer Berlin/Heidelberg, 2008.

[18] A. Boldyreva, N. Chenette, Y. Lee, and A. O'neill. Order-preserving symmetric encryption. In Advances in Cryptology-EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings 28, pages 224–241. Springer, 2009.

[19] A. Boldyreva, S. Fehr, and A. O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In Advances in Cryptology–CRYPTO 2008: 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings 28, pages 335–359. Springer, 2008.

[20] K. Cai, X. Xiao, and G. Cormode. Privlava: synthesizing relational data with foreign keys under differential privacy. Proceedings of the ACM on Management of Data, 1(2):1–25, 2023.

[21] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In Proceedings 42nd IEEE Symposium on Foundations of Computer Science, pages 136–145. IEEE, 2001.

[22] T.-H. H. Chan, K.-M. Chung, B. Maggs, and E. Shi. Foundations of differentially oblivious algorithms. ACM Journal of the ACM (JACM), 69(4):1–49, 2022.

[23] Z. Chang, D. Xie, S. Wang, and F. Li. Towards practical oblivious join. In Proceedings of the 2022 International Conference on Management of Data, pages 803–817, 2022.

[24] J. Chicaiza, M. C. Cabrera-Loayza, R. Elizalde, and N. Piedra. Application of data anonymization in learning

analytics. In Proceedings of the 3rd International Conference on Applications of Intelligent Systems, pages 1–6, 2020.

[25] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. Journal of the ACM (JACM), 45(6):965–981, 1998.

[26] A. R. Chowdhury, C. Wang, X. He, A. Machanavajjhala, and S. Jha. Cryptϵ: Crypto-assisted differential privacy on untrusted servers. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pages 603–619, 2020.

[27] S. Chu, D. Zhuo, E. Shi, and T. H. Chan. Differentially oblivious database joins: Overcoming the worst-case curse of fully oblivious algorithms. In 2nd Conference on Information-Theoretic Cryptography, 2021.

[28] G. Cormode, S. Jha, T. Kulkarni, N. Li, D. Srivastava, and T. Wang. Privacy at scale: Local differential privacy in practice. In Proceedings of the 2018 International Conference on Management of Data, pages 1655–1658, 2018.

[29] N. Crooks, M. Burke, E. Cecchetti, S. Harel, R. Agarwal, and L. Alvisi. Obladi: Oblivious serializable transactions in the cloud. In 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), pages 727–743, 2018.

[30] E. Dauterman, V. Fang, I. Demertzis, N. Crooks, and R. A. Popa. Snoopy: Surpassing the scalability bottleneck of oblivious storage. In Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles, pages 655–671, 2021.

[31] A. Dave, C. Leung, R. A. Popa, J. E. Gonzalez, and I. Stoica. Oblivious coopetitive analytics using hardware enclaves. In Proceedings of the Fifteenth European Conference on Computer Systems, pages 1–17, 2020.

[32] I. Demertzis, D. Papadopoulos, C. Papamanthou, and S. Shintre. {SEAL}: Attack mitigation for encrypted databases via adjustable leakage. In 29th USENIX security symposium (USENIX Security 20), pages 2433–2450, 2020.

[33] T. Dinh Ngoc, B. Bui, S. Bitchebe, A. Tchana, V. Schiavoni, P. Felber, and D. Hagimont. Everything you should know about intel sgx performance on virtualized systems. Proceedings of the ACM on Measurement and Analysis of Computing Systems, 3(1):1–21, 2019.

[34] C. Dwork. Differential Privacy. In Automata, Languages and Programming. Springer, 2006.

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

[36] C. Dwork, N. Kohli, and D. Mulligan. Differential privacy in practice: Expose your epsilons! Journal of Privacy and Confidentiality, 9(2), 2019.

[37] M. El-Hindi, C. Binnig, A. Arasu, D. Kossmann, and R. Ramamurthy. Blockchaindb: A shared database on blockchains. Proceedings of the VLDB Endowment, 12(11):1597–1609, 2019.

[38] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information theory, 31(4):469–472, 1985.

[39] S. Eskandarian and M. Zaharia. Oblidb: oblivious query processing for secure databases. Proceedings of the VLDB Endowment, 13(2):169–183, 2019.

[40] C. Ge, S. Mohapatra, X. He, and I. F. Ilyas. Kamino: constraint-aware differentially private data synthesis. Proceedings of the VLDB Endowment, 14(10):1886–1899, 2021.

[41] C. Gentry. Fully homomorphic encryption using ideal lattices. In Proceedings of the forty-first annual ACM symposium on Theory of computing, pages 169–178, 2009.

[42] D. Giantsidi, M. Bailleu, N. Crooks, and P. Bhatotia. Treaty: Secure distributed transactions. In 2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pages 14–27. IEEE, 2022.

[43] O. Goldreich. Towards a theory of software protection and simulation by oblivious rams. In Proceedings of the nineteenth annual ACM symposium on Theory of computing, pages 182–194, 1987.

[44] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In Proceedings of the nineteenth annual ACM symposium on Theory of computing, pages 218–229, 1987.

[45] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. Journal of the ACM, 38(3):691–729, 1991.

[46] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In Proceedings of the seventeenth annual ACM symposium on Theory of computing, pages 291–304, 1985.

[47] P. Grubbs, A. Khandelwal, M.-S. Lacharité, L. Brown, L. Li, R. Agarwal, and T. Ristenpart. Pancake: Frequency smoothing for encrypted data stores. In 29th USENIX Security Symposium (USENIX Security 20), pages 2451–

2468, 2020.

[48] P. Grubbs, R. McPherson, M. Naveed, T. Ristenpart, and V. Shmatikov. Breaking web applications built on top of encrypted data. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pages 1353–1364, 2016.

[49] X. Gu, M. Li, Y. Cao, and L. Xiong. Supporting both range queries and frequency estimation with local differential privacy. In 2019 IEEE Conference on Communications and Network Security (CNS), pages 124–132. IEEE, 2019.

[50] P. Gupta, Y. Li, S. Mehrotra, N. Panwar, S. Sharma, and S. Almanee. Obscure: Information-theoretic oblivious and verifiable aggregation queries. Proceedings of the VLDB Endowment, 12(9), 2019.

[51] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra. Executing sql over encrypted data in the database-service-provider model. In Proceedings of the 2002 ACM SIGMOD international conference on Management of data, pages 216–227, 2002.

[52] T. Hackenjos, F. Hahn, and F. Kerschbaum. Sagma: secure aggregation grouped by multiple attributes. In Proceedings of the 2020 ACM SIGMOD international conference on management of data, pages 587–601, 2020.

[53] M. Hastings, B. Hemenway, D. Noble, and S. Zdancewic. Sok: General purpose compilers for secure multi-party computation. In 2019 IEEE Symposium on Security and Privacy (SP), pages 1220–1237, 2019.

[54] X. He, A. Machanavajjhala, C. Flynn, and D. Srivastava. Composing differential privacy and secure computation: A case study on scaling private record linkage. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 1389–1406, 2017.

[55] Z. He, W. K. Wong, B. Kao, D. W. L. Cheung, R. Li, S. M. Yiu, and E. Lo. Sdb: A secure query processing system with data interoperability. Proceedings of the VLDB Endowment, 8(12):1876–1879, 2015.

[56] N. Johnson, J. P. Near, J. M. Hellerstein, and D. Song. Chorus: a programming framework for building scalable differential privacy mechanisms. In 2020 IEEE European Symposium on Security and Privacy (EuroS&P), pages 535–551. IEEE, 2020.

[57] N. Johnson, J. P. Near, and D. Song. Towards practical differential privacy for sql queries. Proceedings of the VLDB Endowment, 11(5), 2018.

[58] G. Kellaris, G. Kollios, K. Nissim, and A. O'neill. Generic attacks on secure outsourced databases. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pages 1329–1340, 2016.

[59] G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill. Accessing data while preserving privacy. CoRR, abs/1706.01552, 5, 2017.

[60] D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, pages 193–204, 2011.

[61] I. Kotsogiannis, Y. Tao, X. He, M. Fanaeepour, A. Machanavajjhala, M. Hay, and G. Miklau. Privatesql: a differentially private sql query engine. Proceedings of the VLDB Endowment, 12(11):1371–1384, 2019.

[62] I. Kotsogiannis, Y. Tao, A. Machanavajjhala, G. Miklau, and M. Hay. Architecting a differentially private sql engine. In CIDR, 2019.

[63] S. Krastnikov, F. Kerschbaum, and D. Stebila. Efficient oblivious database joins. Proceedings of the VLDB Endowment, 13(11), 2020.

[64] J. Lee and C. Clifton. How much is enough? choosing ε for differential privacy. In Information Security: 14th International Conference, ISC 2011, Xi'an, China, October 26-29, 2011. Proceedings 14, pages 325–340. Springer, 2011.

[65] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pages 49–60, 2005.

[66] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In 2007 IEEE 23rd international conference on data engineering, pages 106–115. IEEE, 2006.

[67] S. Li, Y. Zeng, Y. Wang, Y. Zhong, Z. Zhou, and Y. Tong. An experimental study on federated equi-joins. IEEE Transactions on Knowledge and Data Engineering, 2024.

[68] X. Li, N. Sun, Y. Luo, and M. Gao. Soda: A set of fast oblivious algorithms in distributed secure data analytics. Proceedings of the VLDB Endowment, 16(7):1671–1684, 2023.

[69] X. Li, G. Tan, X. Wang, J. Rogers, and S. Homsi. RESCU-SQL: Oblivious querying for the zero trust cloud. Proceedings of the VLDB Endowment, 16(12):4086–4089, 2023.

[70] X. Li, C. Weng, Y. Xu, X. Wang, and J. Rogers. ZKSQL: Verifiable and efficient query evaluation with zero-knowledge proofs. Proceedings of the VLDB Endowment, 16(8):1804–1816, 2023.

[71] Y. Li and M. Chen. Privacy preserving joins. In 2008 IEEE 24th International Conference on Data Engineering, pages 1352–1354. IEEE, 2008.

[72] Y. Li, Y. Liu, B. Li, W. Wang, and N. Liu. Towards practical differential privacy in data analysis: Understanding the effect of epsilon on utility in private erm. Computers & Security, 128:103147, 2023.

[73] J. Liagouris, V. Kalavri, M. Faisal, and M. Varia. Secrecy: Secure collaborative analytics in untrusted clouds. In 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23), pages 1031–1056, 2023.

[74] Y. Lindell. Secure multiparty computation. Commun. ACM, 64(1):86–96, 2021.

[75] J. R. Lorch, B. Parno, J. Mickens, M. Raykova, and J. Schiffman. Shroud: Ensuring private access to large-scale data in the data center. In 11th USENIX Conference on File and Storage Technologies (FAST 13), pages 199–213, 2013.

[76] Q. Luo, Y. Wang, W. Dong, and K. Yi. Secure query processing with linear complexity. arXiv preprint arXiv:2403.13492, 2024.

[77] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In 2008 IEEE 24th international conference on data engineering, pages 277–286. IEEE, 2008.

[78] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. Acm transactions on knowledge discovery from data (tkdd), 1(1):3–es, 2007.

[79] S. Maiyya, S. C. Vemula, D. Agrawal, A. E. Abbadi, and F. Kerschbaum. Waffle: An online oblivious datastore for protecting data access patterns. Proceedings of the ACM on Management of Data, 1(4):1–25, 2023.

[80] E. A. Markatou and R. Tamassia. Full database reconstruction with access and search pattern leakage. In International Conference on Information Security, pages 25–43, 2019.

[81] F. D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, pages 19–30, 2009.

[82] M. Mirabi and C. Binnig. Qfilter: Towards a fine-grained access control for aggregation query processing over secret shared data. Proceedings of the VLDB Endowment, 16(8):2005–2023, 2023.

[83] I. Mironov, O. Pandey, O. Reingold, and S. Vadhan. Computational differential privacy. In Annual International Cryptology Conference, pages 126–142. Springer, 2009.

[84] P. Mohassel, P. Rindal, and M. Rosulek. Fast database joins and PSI for secret shared data. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pages 1271–1287, 2020.

[85] A. Narayan and A. Haeberlen. {DJoin}: Differentially private join queries over distributed databases. In 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12), pages 149–162, 2012.

[86] A. Narayanan and E. W. Felten. No silver bullet: De-identification still doesn't work. White Paper, 8, 2014.

[87] M. Naveed, S. Kamara, and C. V. Wright. Inference attacks on property-preserving encrypted databases. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pages 644–655, 2015.

[88] J. P. Near, D. Darais, N. Lefkovitz, G. Howarth, et al. Guidelines for evaluating differential privacy guarantees. Technical report, National Institute of Standards and Technology, 2023.

[89] M. E. Nergiz, C. Clifton, and A. E. Nergiz. Multirelational k-anonymity. IEEE Transactions on Knowledge and Data Engineering, 21(8):1104–1117, 2008.

[90] Office for Civil Rights, Health and Human Services. Standards for privacy of individually identifiable health information. Federal Register, 67(157):53181, 2002.

[91] P. Ohm. Broken promises of privacy: Responding to the surprising failure of anonymization. UCLA l. Rev., 57:1701, 2009.

[92] F. Olumofin and I. Goldberg. Privacy-preserving queries over relational databases. In Privacy Enhancing Technologies: 10th International Symposium, PETS 2010, Berlin, Germany, July 21-23, 2010. Proceedings 10, pages 75–92. Springer, 2010.

[93] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In International conference on the theory and applications of cryptographic techniques, pages 223–238. Springer, 1999.

[94] Y. Peng, M. Du, F. Li, R. Cheng, and D. Song. Falcondb: Blockchain-based collaborative database. In Proceedings of the 2020 ACM SIGMOD international conference on management of data, pages 637–652, 2020.

[95] S. Pinto and N. Santos. Demystifying arm trustzone: A comprehensive survey. ACM computing surveys (CSUR), 51(6):1–36, 2019.

[96] R. Poddar, S. Kalra, A. Yanai, R. Deng, R. A. Popa, and J. M. Hellerstein. Senate: a maliciously-secure mpc platform for collaborative analytics. In 30th USENIX Security Symposium (USENIX Security 21), pages 2129–2146, 2021.

[97] R. A. Popa, C. M. Redfield, N. Zeldovich, and H. Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In Proceedings of the twenty-third ACM symposium on operating systems principles, pages 85–100, 2011.

[98] C. Priebe, K. Vaswani, and M. Costa. Enclavedb: A secure database using sgx. In 2018 IEEE Symposium on Security and Privacy (SP), pages 264–278. IEEE, 2018.

[99] D. Proserpio, S. Goldberg, and F. McSherry. Calibrating data to sensitivity in private data analysis. Proceedings of the VLDB Endowment, 7(8), 2014.

[100] L. Qin, R. Jayaram, E. Shi, Z. Song, D. Zhuo, and S. Chu. Adore: Differentially oblivious relational database operators. Proceedings of the VLDB Endowment, 16(4):842–855, 2022.

[101] L. Qiu, G. Kellaris, N. Mamoulis, K. Nissim, and G. Kollios. Doquet: Differentially oblivious range and join queries with private data structures. Proceedings of the VLDB Endowment, 16(13):4160–4173, 2023.

[102] S. Raghuraman and P. Rindal. Blazing fast PSI from improved OKVS and subfield VOLE. In Proceedings of the 2022 ACM SIGSAC Conference. ACM, 2022.

[103] L. Ren, C. Fletcher, A. Kwon, E. Stefanov, E. Shi, M. Van Dijk, and S. Devadas. Constants count: Practical improvements to oblivious {RAM}. In 24th USENIX Security Symposium (USENIX Security 15), pages 415–430, 2015.

[104] X. Ren, L. Su, Z. Gu, S. Wang, F. Li, Y. Xie, S. Bian, C. Li, and F. Zhang. Heda: multi-attribute unbounded aggregation over homomorphically encrypted database. Proceedings of the VLDB Endowment, 2022.

[105] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2):120–126, 1978.

[106] L. Rocher, J. M. Hendrickx, and Y.-A. De Montjoye. Estimating the success of re-identifications in incomplete datasets using generative models. Nature communications, 10(1):1–9, 2019.

[107] J. Rogers, E. Adetoro, J. Bater, T. Canter, D. Fu, A. Hamilton, A. Hassan, A. Martinez, E. Michalski, V. Mitrovic, et al. Vaultdb: A real-world pilot of secure multi-party computation within a clinical research network. arXiv preprint arXiv:2203.00146, 2022.

[108] E. Roth, H. Zhang, A. Haeberlen, and B. C. Pierce. Orchard: Differentially private analytics at scale. In 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20), pages 1065–1081, 2020.

[109] I. Roy, S. T. Setty, A. Kilzer, V. Shmatikov, and E. Witchel. Airavat: Security and privacy for mapreduce. In Usenix Org, pages 297–312, 2011.

[110] C. Sahin, V. Zakhary, A. E. Abbadi, H. Lin, and S. Tessaro. Taostore: Overcoming asynchronicity in oblivious data storage. In 2016 IEEE Symposium on Security and Privacy (SP), pages 198–217. IEEE, 2016.

[111] M. Seastrom. Best Practices for Determining Subgroup Size in Accountability Systems While Protecting Personally Identifiable Student Information. Institute of Education Sciences, IES 2017(147), 2017.

[112] D. Sohn, K. Jiang, and J. Rogers. Alchemy: An optimizer for oblivious sql queries, 2023.

[113] E. Stefanov and E. Shi. Oblivistore: High performance oblivious cloud storage. In 2013 IEEE Symposium on Security and Privacy, pages 253–267. IEEE, 2013.

[114] E. Stefanov, M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. devadas. Path oram: An extremely simple oblivious ram protocol. In Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security (CCS), pages 299–310, 2013.

[115] L. Sweeney. k-anonymity: A model for protecting privacy. International journal of uncertainty, fuzziness and knowledge-based systems, 10(05):557–570, 2002.

[116] Y. Tong, X. Pan, Y. Zeng, Y. Shi, C. Xue, Z. Zhou, X. Zhang, L. Chen, Y. Xu, and e. a. Ke Xu. Hu-fu: Efficient and secure spatial queries over data federation. Proceedings of the VLDB Endowment, 15(6):1159, 2022.

[117] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich. Processing analytical queries over encrypted data. Proceedings of the VLDB Endowment, 6(5), 2013.

[118] D. Vinayagamurthy, A. Gribov, and S. Gorbunov. Stealthdb: a scalable encrypted database with full sql query support. Proceedings on Privacy Enhancing Technologies, 2019.

[119] N. Volgushev, M. Schwarzkopf, B. Getchell, M. Varia, A. Lapets, and A. Bestavros. Conclave: secure multi-party computation on big data. In Proceedings of the Fourteenth EuroSys Conference 2019, pages 1–18, 2019.

[120] S. Wagh, P. Cuff, and P. Mittal. Differentially private oblivious ram. Proceedings on Privacy Enhancing Technologies, 2018(4):64–84, 2018.

[121] S. Wagh, X. He, A. Machanavajjhala, and P. Mittal. Dp-cryptography: marrying differential privacy and cryptography

in emerging applications. Communications of the ACM, 64(2):84–93, 2021.

[122] C. Wang, J. Bater, K. Nayak, and A. Machanavajjhala. DP-sync: Hiding update patterns in secure outsourced databases with differential privacy. In Proceedings of the 2021 International Conference on Management of Data, pages 1892–1905, 2021.

[123] C. Wang, J. Bater, K. Nayak, and A. Machanavajjhala. Incshrink: Architecting efficient outsourced databases using incremental mpc and differential privacy. In SIGMOD'22: Proceedings of the 2022 International Conference on Management of Data, 2022.

[124] F. Wang, C. Yun, S. Goldwasser, V. Vaikuntanathan, and M. Zaharia. Splinter: Practical private queries on public data. In 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), pages 299–313, 2017.

[125] T. Wang, J. Blocki, N. Li, and S. Jha. Locally differentially private protocols for frequency estimation. In 26th USENIX Security Symposium (USENIX Security 17), pages 729–745, 2017.

[126] Y. Wang, Z. Ding, Y. Xiao, D. Kifer, and D. Zhang. Dpgen: Automated program synthesis for differential privacy. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pages 393–411, 2021.

[127] Y. Wang and K. Yi. Secure yannakakis: Join-aggregate queries over private data. In Proceedings of the 2021 International Conference on Management of Data, pages 1969–1981, 2021.

[128] W. K. Wong, B. Kao, D. W. L. Cheung, R. Li, and S. M. Yiu. Secure query processing with data interoperability in a cloud database environment. In Proceedings of the 2014 ACM SIGMOD international conference on Management of data, pages 1395–1406, 2014.

[129] A. Wood, M. Altman, A. Bembenek, M. Bun, M. Gaboardi, J. Honaker, K. Nissim, D. R. O'Brien, T. Steinke, and S. Vadhan. Differential privacy: A primer for a non-technical audience. Vanderbilt Journal of Entertainment & Tech Law, 21:209, 2018.

[130] X. Xiong, S. Liu, D. Li, Z. Cai, and X. Niu. A comprehensive survey on local differential privacy. Security and Communication Networks, 2020(1):8829523, 2020.

[131] M. Xu, B. Ding, T. Wang, and J. Zhou. Collecting and analyzing data jointly from multiple services under local differential privacy. Proceedings of the VLDB Endowment, 13(11), 2013.

[132] C. Yue, T. T. A. Dinh, Z. Xie, M. Zhang, G. Chen, B. C. Ooi, and X. Xiao. Glassdb: An efficient verifiable ledger database system through transparency. VLDB, page 1359–1371, 2023.

[133] D. Zhang, R. McKenna, I. Kotsogiannis, M. Hay, A. Machanavajjhala, and G. Miklau. Ektelo: A framework for defining differentially-private computations. In Proceedings of the 2018 International Conference on Management of Data, pages 115–130, 2018.

[134] Y. Zhang, J. Bater, K. Nayak, and A. Machanavajjhala. Longshot: Indexing growing databases using mpc and differential privacy. In Proceedings of the VLDB Endowment, pages 2005–2018, 2023.

[135] Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou. vsql: Verifying arbitrary sql queries over dynamic outsourced databases. In 2017 IEEE Symposium on Security and Privacy (SP), pages 863–880, 2017.

[136] Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou. A zero-knowledge version of vsql. Cryptology ePrint Archive, Paper 2017/1146, 2017. `https://eprint.iacr.org/2017/1146`.

[137] Y. Zhang, J. Katz, and C. Papamanthou. Integridb: Verifiable sql for outsourced databases. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pages 1480–1491, 2015.

[138] W. Zheng, A. Dave, J. G. Beekman, R. A. Popa, J. E. Gonzalez, and I. Stoica. Opaque: An oblivious and encrypted distributed analytics platform. In 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), pages 283–298, 2017.

[139] M. Zhou, E. Shi, T.-H. H. Chan, and S. Maimon. A theory of composition for differential obliviousness. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 3–34. Springer, 2023.

[140] W. Zhou, Y. Cai, Y. Peng, S. Wang, K. Ma, and F. Li. Veridb: An SGX-based verifiable database. In Proceedings of the 2021 International Conference on Management of Data, pages 2182–2194, 2021.

Differential Privacy with Fine-Grained Provenance: Opportunities and Challenges

Xi He
University of Waterloo
xi.he@uwaterloo.ca

Shufan Zhang
University of Waterloo
shufan.zhang@uwaterloo.ca

Abstract

Differential privacy (DP) offers a robust framework for protecting individual privacy when analyzing data. However, the elegant abstractions used in DP theory do not always translate seamlessly to real-world systems. For example, the basic DP system maintains a global privacy budget and updates it when it answers a query. However, tracking this level of privacy information cannot character the privacy loss for heterogeneous data types, complex DP mechanisms, or the privacy loss towards different data analysts. Several DP systems have shown success by tracking more fine-grained information about privacy usage. Inspired by this, we propose a novel perspective: leveraging fine-grained privacy provenance to build practical DP systems. First, we propose a taxonomy of privacy provenance for DP, including why-, how-, and where-DP-provenance that characterizes different aspects of DP. Next, we review several systems that feature different techniques in each category of privacy provenance. Through this unique lens, we summarize the open challenges for DP systems and future directions for deeper integration between DP and data provenance techniques.

1 Introduction

Differential privacy (DP) [40] emerged in 2006 as a groundbreaking concept for protecting individual privacy in data analysis. DP offers a powerful privacy-preserving approach by mathematically ensuring that data releases reveal minimal information about any single person. This has led to the development of numerous DP mechanisms and systems like PINQ [88], FLEX [65], PrivateSQL [75], GoogleDP [5], and Chorus [66]. However, despite its theoretical elegance and strong privacy guarantees, DP's practical deployments lag behind its potential. While a few pioneering cases exist, such as the 2020 US Census disclosure [1, 57], widespread adoption remains limited. Companies like Amazon, Snowflake, Google, LinkedIn, Uber, and Apple, and startups like Tumlut Labs and Transcend are exploring DP in data management products or statistical learning scenarios [4, 111, 53, 128, 60, 107, 27, 96, 118, 117], but these integrations are often experimental and face challenges in production environments [114, 47, 134], suggesting a divergence between theory and practice.

In theory, DP relies on a privacy budget, represented by the parameters (ϵ, δ), which controls the overall privacy guarantee. By carefully injecting controllable noise, it can be proved that a mechanism can only reveal bounded information about any individual in a *static* dataset, and thus, this mechanism satisfies the notion of DP.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

However, translating this theory into practical systems presents several challenges. First, even with a simple use case that only focuses on the central DP setting[1], the system has to interact with private data and data analysts and has to maintain the system budget at least correctly and faithfully. Some systems resort to large or frequently reset budgets [107, 4], which may jeopardize long-term privacy. Second, DP systems often assume static datasets. However, real-world data can be dynamic, integrating information from various sources [78] and undergoing regular updates [22]. The individuals in the dataset may have different privacy awareness [67]. These complexities require additional considerations to maintain privacy guarantees. Third, DP systems need to cater to analysts with varying levels of privacy expertise. Non-expert analysts may struggle to interpret noisy results or choose the most appropriate DP mechanism for their queries [48], especially for complex queries, like nested subqueries or batched workloads [95, 29]. As a result, implementations of the DP system can fail to deliver an optimized privacy-utility trade-off or the expected privacy guarantees.

Recent DP systems have addressed one specific challenge mentioned above by tracking fine-grained information such as the data blocks [80, 78] or the noise used for previous queries [87, 137]. Inspired by these works, we propose a broader approach to building usable end-to-end DP systems by leveraging the data provenance framework in databases [18], in which tracing and propagating *proper provenance metadata* turns out to be useful for understanding queries, integrating data, and debugging inconsistencies. Similarly, in DP systems, we envision privacy provenance — metadata that tracks the DP mechanisms and benefits the users of the systems.

In this work, we analyze different components of a DP system and explore how proper provenance metadata can offer benefits, including improved user understanding, enhanced utility optimization, and dynamic privacy management. We start with three types of privacy provenance: *why-DP-provenance*, which explains the private/noisy outputs to the data analysts; *how-DP-provenance*, which uses metadata for tighter privacy in running DP mechanisms; and *where-DP-provenance*, which tracks privacy budget consumption over dynamic data sources to satisfy different resolutions of privacy definitions. While the concept of privacy provenance is a recent development [138], the idea of leveraging additional data structures in DP algorithm design has been explored in prior work [66, 88, 46, 90]. We surveyed all the relevant DP works in our privacy provenance framework and provided discussion in this direction. Note that our characterization of privacy provenance in this work is not meant to be exhaustive. We hope this work stimulates further exploration and discussion in developing more usable and optimized systems for DP. This work also aims to complement existing visions on DP (including recent surveys [23, 92]). By introducing a system-oriented perspective through the lens of privacy provenance, we hope to pave the way for the development of more usable and optimized DP systems in the future.

Article Roadmap. The remainder of this article is organized as follows. Section 2 summarizes the preliminaries of differential privacy and provenance in databases. In Section 3, we provide a systematic view of the complexity of the DP systems and propose a taxonomy of fine-grained privacy provenance. We survey several systems that feature the usages of fine-grained privacy provenance in Section 4 and discuss challenges and future directions in Section 5. We conclude this article in Section 6.

2 Background

We introduce and summarize the related definitions of differential privacy. Next, we introduce the necessary preliminaries for differential privacy and provenance in databases.

2.1 Definition of Differential Privacy

Definition 2.1: *[Differential Privacy [40]]*

[1]Assuming the existence of a trusted curator runs a (data analytics) system with DP guarantees for a curated sensitive dataset.

We say that a randomized algorithm $\mathcal{M} :\to \mathcal{O}$ satisfies (ϵ, δ)-differential privacy (DP), if for any two neighbouring databases (D, D') that differ in only 1 tuple, and $O \subseteq \mathcal{O}$, we have

$$\Pr[\mathcal{M}(D) \in O] \le e^{\epsilon} \Pr[\mathcal{M}(D') \in O] + \delta.$$

Definition 2.2: *[Global Sensitivity] For a query $q : \mathcal{D} \to \mathbb{R}^d$ the ℓ_2 global sensitivity of this query is*

$$\Delta q = \max_{D,D':d(D,D')\le 1} \|q(D) - q(D')\|_2,$$

where $d(\cdot,\cdot)$ denotes the number of tuples that D and D' differ and $\|\cdot\|_2$ denotes the ℓ_2 norm. If we replace the ℓ_2 norm with ℓ_1 norm, then we obtain the ℓ_1 sensitivity $\Delta_1 q$.

Basic DP Mechanisms. The most basic DP mechanisms are the Laplace and Gaussian mechanisms, injecting Laplace and Gaussian noises, respectively, into the query answers, which are explained below.

Definition 2.3: *[Laplace Mechanism [39]] Given a numerical query $q : \mathcal{D} \to \mathbb{R}^d$, the Laplace mechanism outputs $\mathcal{M}(D) = q(D) + \eta$ where $\eta \sim Lap\,(b)^d$ where $Lap\,(b)^d$ is a vector of d i.i.d. samples from a Laplace distribution with scale b. If $b = \Delta_1 q/\epsilon$, then the Laplace mechanism preserves $(\epsilon, 0)$-DP.*

Definition 2.4: *[Gaussian Mechanism [39]] Let $\epsilon \in (0,1)$. Given a numerical query $q : \mathcal{D} \to \mathbb{R}^d$, for constant $c > \sqrt{2\ln(1.25/\delta)}$, the Gaussian mechanism adds the noise vector $(\eta_1, \eta_2, \ldots, \eta_d)$ to the query answer $q(D)$, where η_i are i.i.d. random variables drawn from the Gaussian distribution $\mathcal{N}(0, \sigma^2 I)$ with $\sigma > c\Delta q/\epsilon$. The Gaussian mechanism is (ϵ, δ)-differentially private.*

The standard Gaussian mechanism [39] has the limitation that it can only be used in a high privacy regime, where the privacy parameter ϵ should be within the range of $(0, 1)$. Balle and Wang [6] propose an improved mechanism, namely the analytic Gaussian mechanism, overcoming this limitation in the standard Gaussian mechanism. We give the definition of analytic Gaussian mechanism below for completeness, while skipping the details of the mechanism does not affect understanding this article.

Definition 2.5: *[Analytic Gaussian Mechanism [6]] Given a query $q : \mathcal{D} \to \mathbb{R}^d$, the analytic Gaussian mechanism $\mathcal{M}(D) = q(D) + \eta$ where $\eta \sim \mathcal{N}\left(0, \sigma^2 I\right)$ is (ϵ, δ)-DP if and only if*

$$\Phi_{\mathcal{N}}\left(\frac{\Delta q}{2\sigma} - \frac{\epsilon\sigma}{\Delta q}\right) - e^{\epsilon}\Phi_{\mathcal{N}}\left(-\frac{\Delta q}{2\sigma} - \frac{\epsilon\sigma}{\Delta q}\right) \le \delta,$$

where $\Phi_{\mathcal{N}}$ denotes the cumulative density function (CDF) of Gaussian distribution. In this mechanism, the Gaussian variance is determined by $\sigma = \alpha\Delta q/\sqrt{2\epsilon}$ where α is a parameter determined by ϵ and δ [6].

Privacy Composition Theory. Differential privacy enjoys the nice property of being compositional. Running a DP mechanism multiple times is also DP, but with a higher privacy cost. One can, therefore, build complex mechanisms by composing basic DP mechanisms using the following composition theorems.

Theorem 2.1: *[Sequential Composition [39]] Given two mechanisms $\mathcal{M}_1 :\to \mathcal{O}_1$ and $\mathcal{M}_2 :\to \mathcal{O}_2$, such that $\mathcal{M}_1$ satisfies (ϵ_1, δ_1)-DP and $\mathcal{M}_2$ satisfies (ϵ_2, δ_2)-DP. The combination of the two mechanisms $\mathcal{M}_{1,2} :\to \mathcal{O}_1 \times \mathcal{O}_2$, which is a mapping $\mathcal{M}_{1,2}(D) = (\mathcal{M}_1(D), \mathcal{M}_2(D))$, is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DP.*

Theorem 2.2: *[Parallel Composition [88]] Let a mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{O}$ be (ϵ, δ)-DP. If $D_1, \ldots, D_n \in \mathcal{D}$ are n arbitrary disjoint sets of databases and $X = D_1 \cup \cdots \cup D_n$, the release of mechanism output sequence $\mathcal{M}(D_1), \ldots, \mathcal{M}(D_n)$ satisfies (ϵ, δ)-DP.*

Theorem 2.3: *[Post Processing [39]] For an (ϵ, δ)-DP mechanism $\mathcal{M}$, applying any arbitrary function f over the output of $\mathcal{M}$, that is, the composed mechanism $f \circ \mathcal{M}$, satisfies (ϵ, δ)-DP.*

2.2 Provenance in Databases

Provenance (sometimes also called lineage or pedigree) in databases [18] studies the origin or history of the data through a high-level structured computation (e.g., data transformation or query execution) in its lifecycle. By recording more metadata through annotation [15] or additional data structure (e.g., semiring) [55, 54], data management systems can explain the relationships on how data items in the output (e.g., a tuple produced) depend on (various) source/input (e.g., multiple relational tables). The most common provenance models are "why-", "how-", and "where-" provenances, aim to explain, respectively, why a particular tuple appears in the output [14], how an output tuple is processed and derived [55], and where, precisely, an output tuple is originated in the sources according to the computation [14]. Provenance information is useful for understanding the query behavior, data processing steps, and scientific workflows, and thus critical in auditing the integrity, reproducibility, and reliability of data in many scenarios [18].

The relationship between provenance and data privacy has received increasing attention [97, 25, 24, 26, 98, 99]. This line of research focuses on the potential privacy risks associated with tracking provenance information, particularly for sensitive data. Researchers explore how revealing provenance details could leak information and investigate techniques to minimize such risks. This might involve suppressing specific parts of provenance queries or hiding certain intermediate data while still achieving desired privacy guarantees. Other recent work [130] highlights how provenance can be used for maintaining and versioning machine learning models to mitigate privacy attacks towards the models.

3 Privacy Provenance Overview

This section explores how provenance information can be leveraged to enhance the effectiveness of DP systems. We introduce the concept of DP provenance, a systematic approach to capture and utilize provenance to support DP systems. We propose a new taxonomy that categorizes existing work based on the type and granularity of the provenance considered. We then delve into how different types of provenance can be used to improve various aspects of DP systems.

3.1 DP System Model: A Provenance Perspective

Every data system that integrates DP has to record and track some additional metadata, compared to that without DP. We start with a simple, hypothetical system with DP to show the most basic functionalities and the coarsest privacy provenance tracking provided by it. We then lay out the different entities/components of a DP system that motivate the need for finer-grained privacy provenance or additional provenance features.

A (Hypothetically) Basic DP System. Consider running a DP system in the central setting, where a trusted data curator maintains a protected database D. The data curator sets up a finite system-wise or global privacy budget to bound the overall extent of information disclosure over this database. (A group of) untrusted data analyst(s) would like to query the private database D. Each incoming query from a data analyst specifies a per-query privacy budget that indicates the amount of budget they would like to spend on this particular query. The DP system uses *a fixed mechanism* (e.g., the Laplace mechanism) to answer this query, and subtracts the per-query privacy budget from the global budget. The system rejects a query if the remaining global budget is not sufficient for this query; it stops processing queries once the global privacy budget is fully depleted.

This basic system can only support limited queries (that could be naïvely answered through one fixed mechanism). Almost every DP system that researchers implement in literature is more complicated than it. Even though, the basic system has to track the privacy loss in terms of the per-query privacy budget and the global budget consumption[2], which is the simplest example of privacy provenance. This oversimplified setting overlooks

[2]Indeed, in real-world implementations which lack careful management, DP can rapidly become excessively restrictive so that service

the complexity of a real-world DP system. Next, we will provide a view of the complexity of system designs for real-world DP systems.

The Complexity of DP Systems. The design of a DP system greatly depends on the users's role, interest, and expertise levels in this system. First, the data analysts (who are the queries or the programmers) of a DP system have no direct access to the data. They care for the accuracy of the query results and how many queries they can interact with the system. Ideally, if the data analysts have the full domain knowledge of DP and how the system works, then they can understand the DP programs supported by the systems and trace the necessary meta information for optimizing their interested queries. If not, they may not be able to specify the privacy budget correctly and understand the noisy outputs. A usable DP system in practice is expected to accommodate the needs of both types of data analysts. It should explain the noisy output to DP novices and provide modularized APIs for DP experts to program their own tasks.

Second, the data curators are responsible for setting up the data input for the programs and allocating privacy budgets among different analysts. The basic DP system makes several assumptions to simplify the privacy analysis. It assumes that the database is static (i.e., not subject to updating), and each row in the database corresponds to a single individual. In addition, the privacy analysis is rigidly enforced at the entire table level, and the protection is uniform across every row in the database. In real-world use cases, however, the underlying database is often under dynamic changing and has the hetergenous nature that not every part of the data is equally sensitive. For example, new data analysts can keep opt-in, and their data will be merged into the private database when the system is running. The data contributors may have different contributions to the database in terms of the number of rows. They may also have personalized opinions or different privacy awareness regarding the protection of their data. In such scenarios, keeping a single global privacy budget and simple privacy analysis at the table level is not sufficient to provide the desired level of privacy guarantees. In addition, data analysts can have different trust or privilege levels when accessing the system. Tech companies, for example, need to query their users' data for internal applications like anomaly detection. They also consider inviting external researchers with low privilege levels to access the same sensitive data for study through the shared query interface. It is unfair if a data analyst with a low privilege level asks queries with a significant portion of the global budget so that higher privileged analysts have no more budget to consume. Therefore, a real DP system involving multiple data analysts may have a range of design choices to make, from privilege allocation to query/task scheduling to assumptions on whether the analysts can collude, etc.

Third, the DP programs responsible for mapping the utility interests of data analysts and the privacy interests of data curators have a wide range of complexities. Some have a fixed query template with a deterministic sensitivity and hence noise scale (e.g., stability tracking in PINQ [88]); some require dynamic sensitivity analysis (e.g., query rewriting in Chorus [66]); and most systems deal with more than one query [88, 66, 48, 44]. The queries to be supported can be complicated, e.g., involving multiple transformations of data, and be batched into an OLAP workload. The desirable system should support multiple mechanisms (including customized ones) to answer different types of queries and additional algorithms to choose among the mechanisms to optimize the privacy-utility trade-off for the queries (especially for the workloads).

In every stage/aspect of the DP system, we have observed evidence or challenges that the basic DP system or coarse-grained DP provenance cannot address. We envision that a framework with *fine-grained privacy provenance* can help solve these challenges and ensure a more usable and optimized DP system for practical needs, which is described next.

3.2 The Taxonomy of Fine-Grained Privacy Provenance

We draw an analogy to provenance in databases and, similarly, characterize the privacy provenance into three categories: "why-DP-provenance" (or output-provenance), "how-DP-provenance" (or process-provenance), and

providers have to set up a large (or even infinite) global budget, which has been shown to attacks [114].

Table 1: Comparison between different types of provenance in databases [18] and privacy provenance.

	Database Provenances [18]	**Privacy Provenances**
Why-(DP-)Provenance	Identifying sub-instances of the input that "witness" a part of the output	Explaining why a noisy output satisfies accuracy requirements and/or why certain queries are rejected
How-(DP-)Provenance	Providing additional information on how the output tuple is derived, e.g., transformations applied during processing	Tracing query metadata for tighter privacy analysis during executing DP program/mechanisms (that are complex or involving multiple data analysts)
Where-(DP-)Provenance	Pinpointing where an attribute value in the output tuple is exactly copied from	Managing budget consumption for heterogeneous private input data sources (e.g., user-level DP, personalized DP)

"what-DP-provenance" (or input-provenance), based on the information/metadata they track and the aspects they influence a DP system.

Why-DP-Provenance, also known as output-provenance, refers to the additional information that helps data analysts understand the reasoning behind a DP system's output. It focuses on the interaction between the system's results and the analyst's needs. Why-DP-provenance aims to answer questions like, why is a specific privacy budget chosen for the analyst's query, how does the level of noise in the answer achieve the desired accuracy, why is a noisy answer still considered useful, and what factors lead to a query rejection? Benefits of enforcing fine-grained why-DP-provenance include:

- Explainability and Interpretability: Why-DP-provenance details query answers' usefulness and confidence intervals, which allows analysts to understand the trade-offs between privacy guarantees and accuracy.
- Tighter Privacy Composition: Why-DP-provenance facilitates a more accurate understanding of how different queries impact the overall privacy budget.
- Optimal Privacy-Accuracy Trade-Offs: Why-DP-provenance empowers analysts to make informed decisions about the balance between privacy and the usefulness of results.

Techniques for why-DP-provenance device approaches for accuracy-first query specification, error specification, fine-grained budget specification, and others. Unlike the why-provenance in databases that focuses on "why" something happened or exists, the why-DP-provenance answers "why" a specific level of parameters in a DP algorithm is chosen and its impact on results within a DP system.

How-DP-Provenance, also known as process-provenance, focuses on capturing details about the specific privacy mechanisms employed within a DP system. This information becomes crucial for answering complex queries privately, building complex DP mechanisms, or selecting the optimal/best DP mechanisms for a query workload. How-DP-provenance solves the following questions: How is the noise for a private query calculated based on a series of transformations of data? What structural properties of a query can be used for a tighter privacy bound? Can we reuse the noise calibrated to historical queries for answering new queries to save the privacy budget? Fine-grained how-DP-provenance tracking enables:

- Efficient Privacy Analysis: How-DP-provenance tracks the series of modularized transformations or the query structural properties for complex queries to make sensitivity analysis more efficient.
- Better Utility for Query Workloads: How-DP-provenance reuses noise from historical query answers or injects correlated noise to a batch of queries to increase the overall utility for the workload.

- Effective Budget Allocation: How-DP-provenance facilitates the budget allocation across multiple data analysts so that the privacy loss is tightened when analysts collude.

Techniques that support fine-grained how-DP-provenance include transformation tracking and noise tracking. Different from the how-provenance in databases that generates a derivability relationship between output and input tuples, the how-DP-provenance answers "how" a better DP mechanism could be designed or modularized for answering queries.

Where-DP-Provenance, also known as input-provenance, accounts for which (part of) private input data is used for a DP mechanism. The complex data type can involve multiple database tables, and queries over different tables can result in different DP guarantees — tracking which tables are used for what queries becomes essential for multi-relational DP systems. Furthermore, even with a single table, different queries or mechanisms may be interested in different parts of the table — accounting privacy loss at the table-level can waste more privacy budgets. Third, data may be merged at different timestamps or belong to different users. Thereby, where-DP-provenance tracks privacy loss for data by answering questions of what part of the private data is used to answer a particular query, what (group of) users are associated with the data. With where-DP-provenance, the system gains:

- Flexibility with Multi-Resolution Privacy: Where-DP-provenance allows data curators to specify policies with different resolutions of privacy guarantees.

- Continuously Running System: Where-DP-provenance enables the replacement of retired data with new data so that the system can continuously execute.

While the where-provenance in databases pinpoints the exact places that an output tuple is copied from, the where-DP-provenance answers "where" incurs a privacy loss in the private input w.r.t queries.

Next, we will describe the DP techniques that fit into the framework of each why, how, and where-DP-provenance and the different granularities of these techniques.

3.2.1 Techniques for Why-DP-Provenance

Query Specification. Two types of DP query specification exist: privacy-first and accuracy-first. Most (traditional) DP building blocks or systems are privacy-first, meaning that they require the data analysts to specify a privacy budget for their queries or tasks to run. This privacy budget will be deducted from the global privacy budget if the tasks are executed and the noisy answers are returned to the analysts. While this approach is easy to analyze and has many optimal and off-the-shelf mechanisms developed over the years, it may limit the usability of a DP system: the data analysts care about the quality of the query answers, but may not have sufficient expertise to understand the DP mechanisms and the relationship between privacy and the error rate according to the chosen mechanism. Since a DP system cannot release the true query answer, a single noisy output cannot explain how well the black-box mechanism works to privacy novice analysts. In addition, the released answers (or consumed budget) can not be reversed—discarding unsatisfactory query answers simply wastes the global privacy budget.

Another approach for query specification is *accuracy-first* [48, 50, 138, 126, 132, 83, 87, 100], that allows the data analysts to superimpose an accuracy requirement in the query specification, and the system can translate the accuracy requirements into privacy budget and automatically choose the optimal mechanism to answer this query. This approach aims to close the discrepancy between privacy-oriented DP mechanisms and the needs of data analysts for understanding noisy outputs, but many open problems remain in understanding data-dependent accuracy translation. Recent work like DPella [121] and DProvDB [138] support both the privacy- and accuracy-oriented modes.

Error Specification. A line of work, including DPComp [58], Overlook [116], PSI [46], Bittner et al. [12], DPP [64], ViP [90], and DPXPlain [115], aims to provide interfaces for explaining the noisy output to the data analyst or visualizing an optimal privacy-utility trade-off due to the mechanism to help the analysts make informed choices. These tools explore a DP confidence interval of the noisy answer [21, 37, 19, 36, 90, 113] or other statistical metrics on measuring accuracy of the answer as functions of the selected privacy budget [116, 64]. ViP [90] associates the query output with a differentially private randomization interval, indicating the noise bounds of the query result with high confidence. This randomization bound could be either post-processed from a noisy answer if the mechanism is data-independent or computed with a small portion of additional privacy budget from data-dependent mechanisms [21, 37, 19, 36]. DPXPlain [115], on the other hand, start to explore methods of providing additional explanation of aggregation query results, on which whether an unexpected answer is due to the data itself or the randomness introduced by DP.

Budget Specification. The composition of privacy loss through a series of queries can be classified into two levels, based on whether different analysts are distinguished. The coarser level of query composition is to regard all data analysts as a unified entity, and the system sequential composition to account for privacy loss during the execution of the queries. This method is easy to implement, well-suited to different types of (complex) queries, and can explain whether a query is rejected—the privacy budget or the translated privacy budget exceeds the remaining global privacy budget. Sequential composition can also be replaced by a privacy odometer [108, 79] or adaptive composition [127, 129, 68] that gives tighter privacy bound over the queries that are adaptively asked, but they are restricted to simple queries. More fine-grained query composition is to track the privacy loss as per data analysts and per queries they ask [138]. This approach can not only answer more specific questions on *why this particular analyst's query is rejected* but also achieve fairness among multiple data analysts when they are assigned different trust/privilege levels. However, fine-grained tracking requires recording more metadata proportional to the number of data analysts in the system.

Other Techniques for Why-DP-Provenance: Automated DP Proof Generation. Besides explaining noisy output to the data analysts, another line of work investigates the logical representation and execution of a differentially private mechanism/system. They check and verify the privacy properties of a DP program implementation by either generating automated proofs for DP [125, 2, 124, 123, 135, 93, 3] or detecting counterexamples that violate DP guarantees [28, 9, 10, 8] through annotated type systems and static/dynamic analysis of the program execution.

3.2.2 Techniques for How-DP-Provenance

Transformation Tracking. Complex database queries often consist of a series of transformations over the private data, e.g., selection, projection, group-by, join, union, and aggregation. In order to calibrate noise to the query answers, the system needs to analyze the sensitivity of the query. We categorize the approaches to estimate the sensitivity of a query into three levels, from coarse-grained to fine-grained: tracking sensitivity directly, tracking transformation stability, and tracking query structures.

Tracking Sensitivity (Directly). The coarsest level of how-DP-provenance is to directly compute an upper bound of the query sensitivity. For simple queries, the sensitivities are well understood and can be computed easily. However, for complex queries, without making clever use of the properties of the query, the sensitivity could be overestimated or cumbersome to compute.

Tracking Transformation Stability. Existing systems like PINQ [88] and wPINQ [102] keep track of the *transformation stability* to calculate the amount of noise needed for the queries. For a transformation $T : \mathcal{D} \to \mathcal{D}$, it is c-stable if $\forall$ two input databases $D, D' \in \mathcal{D}$, we have $|T(D) \triangle T(D')| \leq c \times |D \triangle D'|$, where $\triangle$ denotes the symmetric difference between two databases, i.e., $D \triangle D' = (D \backslash D') \cup (D' \backslash D)$. For example, common SQL transformations selection, projection, and counting queries have stability of 1, and group-by has a stability of 2. It has been shown that for a ϵ-differentially private mechanism $\mathcal{M}$ and a c-stable transformation series T, the composition $\mathcal{M} \circ T$ will be $(c \cdot \epsilon)$-differentially private.

Tracking Query Structures. Keeping the transformation stability metadata makes it easy to analyze the privacy loss for a series of linear transformations before histogram queries, but may not be sufficient for highly sensitive queries like aggregation and join. The query sensitivity for aggregations could be as large as the size of the domain product (e.g., sum over selected attributes). The approach to handling aggregation queries is to truncate the attribute domain [66, 45, 31] and/or perform aggregations on disjoint subsamples [95, 89, 110, 66], which requires the system to keep track of the truncation information per query and per attribute and the information about how the truncation transfers over queries. Chorus [66] enables a query rewriting and annotation technique, which could be seen as using *how-DP-provenance*, to automatically trace and analyze the query stability and truncated domain when processing aggregation queries. Similarly, the sensitivity of join queries can be even unbounded — changing one tuple in an input table can cause unbounded changes in the join output since this tuple can match an arbitrary number of tuples in another input table. DP mechanisms for answering join queries clip the maximum number of tuples that a tuple can match in a join [75], or add data-dependent noises [95, 65, 29, 30], or a mixture of both [32]. Indeed, the recent residual sensitivity mechanism [29] analyzes the multi-way join topology metadata and traces a smooth upper bound of local sensitivity across this join topology to *efficiently* calculate a *tighter* noise.

Noise Tracking. While the basic DP system does not track the noise added to each query answer (i.e., queries are regarded as independent), a number of recent work [138, 87, 133, 81, 56, 74] inject correlated noise to the answers. They optimize the accuracy of the query results by batching the query workload and adding correlated noise with the workload of one single data analyst [132, 81, 34], or, in a more fine-grained way, maintaining a stateful cache of the historical query answers [87, 74, 75] so that answers to the new queries can reuse the cached noises. The DP caches are extended from answering one data analyst's queries to mitigating privacy loss across multiple data analysts [138, 43, 78, 61] or tight adaptive composition [119, 120] across analysts when multiple analysts ask the same or similar queries. Other multi-analyst systems [132, 133, 73, 103, 104] tracks per-analyst privacy [132] or accuracy [73] constraints to optimize the privacy-accuracy trade-off or fairly answer queries among analysts [103, 104]. Among all the different settings in multi-analyst DP, additional fine-grained requirements or privacy guarantees regarding different analysts' queries are recorded for the DP mechanism designs, which is in line with how-DP-provenance.

3.2.3 Techniques for Where-DP-Provenance

Privacy Resolutions. Depending on the data model and the intended privacy goals to deliver, a DP system can achieve the notion of *event-level DP*, *user-level DP*, multi-resolution DP (defined as per policies) [59, 75] and other extended notions of DP, e.g., personalized user-level DP [67]. Event-level DP assumes that in the private input data, each individual contributes only one record, while it is a special case of user-level DP, which allows each individual to make multiple contributions and reasons about privacy at the user level. In particular, the neighboring database definition is different in the two settings, which changes the way sensitivity is analyzed. Other DP notions, such as Pufferfish privacy [72], Blowfish privacy [59], multi-resolution privacy [75], per-attribute DP[51], Metric DP [17], geo-indistinguishability [134], etc., relax and extend DP to more general settings for example with correlations.

Privacy Accounting. Accounting for privacy loss over the input data is challenging. The basic DP system performs the privacy composition at the table level. However, it is unlikely that every query touches the entire table. Accounting privacy at the table level would waste privacy budgets for the part of data that is not used for computations. More fine-grained privacy accounting considers splitting databases into disjoint blocks and sub-tables, and only the block that is used for answering a query will be deducted for privacy consumption. This approach is called parallel composition or block composition [88, 80]. This has also been extended in recent work [78] that a finer-grained user-level partitioning is available at the column level so that the wasted privacy budget can be minimized. Other than privacy accounting for different parts of the data, the input data can arrive at different timestamps. Based on the DP models used, existing work proposes mechanisms for sliding

Table 2: Taxonomy of Privacy Provenance and Evaluations of Implemented DP Systems. ✓=with, ✗=without, ○=Coarse-Grained, ◐=Moderate-Grained, ●=Fine-Grained. Shaded rows indicate systems surveyed in case studies.

Systems	Why-DP-Provenance			How-DP-Provenance		Where-DP-Provenance	
	Query Spec	Err Spec	Budget Spec	Transformation	Noise (Trackings)	Priv Resolutions	Priv Accountant
Basic	Privacy-First	✗	○ Per Query	○ Sens Only	✗ No	○ Event-DP	○ Table Level
PINQ (2009)	Privacy-First	✗	○ Per Query	◐ Sens+Stability	✗ No	○ Event-DP	◐ Sub-Table Level
ProPer (2015)	Privacy-First	✗	○ Per Query	◐ Sens+Stability	✗ No	● User-DP[†]	◐ Sub-Table Level
PSIΨ (2016)	Privacy-First	◐	◐ Per Query[*]	○ Sens Only	✗ No	○ Event-DP	○ Table Level
ϵKTELO (2018)	Privacy-First	✗	○ Per Query	◐ Sens+Stability	◐ Workload	○ Event-DP	◐ Sub-Table Level
APEx (2019)	Accuracy-First	○	○ Per Query	○ Sens Only	◐ Workload	○ Event-DP	○ Table Level
PrivateSQL (2019)	Privacy-First	✗	○ Per Query	○ Sens Only	● Caching	● Policy-DP	○ Table Level
Sage (2019)	Privacy-First	✗	○ Per Query	○ Sens Only	✗ No	○ Event-DP	◐ Sub-Table Level
Chorus (2020)	Privacy-First	✗	○ Per Query	● Query Struct	✗ No	○ Event-DP	○ Table Level
CacheDP (2022)	Accuracy-First	○	○ Per Query	○ Sens Only	● Caching	○ Event-DP	○ Table Level
ViP (2022)	Privacy-First	●	◐ Per Query[*]	○ Sens Only	✗ No	○ Event-DP	○ Table Level
DProvDB (2024)	Both	○	● Query+Analyst	◐ Sens+Stability	● Caching	○ Event-DP	○ Table Level
Cohere (2024)	Privacy-First	✗	● Query+Analyst	○ Sens Only	◐ Workload	◐ User-DP	● Column Level

[†] ProPer achieves user-level DP with personalized privacy guarantees.
[*] ViP and PSIΨ both support an interface to help analysts split privacy budget across multiple queries. ViP supports more types of error specifications (confidence intervals, quantiles, etc.).

windows [70], the entire streams [41, 35, 33] or with historical data [22].

Table 2 summarizes the techniques and the granularities of the provenances used in some existing DP systems for DP with trusted data curators (i.e., in the central setting). We discuss other cases that enforce local DP or other models of DP in Section 5.

4 Case Studies on Systems with Fine-Grained Privacy Provenance

This section dives into several representative DP systems that utilize fine-grained why-, how-, and where-provenance techniques via case studies. Each case study overviews the related systems, summarizes their provenance techniques, and discusses their strengths and weaknesses.

4.1 Case Study: APEx for Accuracy-Aware DP Data Exploration

APEx prioritizes accuracy in query specification through why-DP-provenance techniques. Additionally, it employs noise tracking for processing exploration workloads, leveraging how-DP-provenance. These features enhance the usability of the basic DP system.

4.1.1 Problem and Technical Brief

APEx is among the first systems to empower data analysts, even those without prior DP expertise, to easily specify private queries. It achieves this by allowing analysts to focus on their desired outcome, the answer's accuracy, rather than needing to grapple with complex privacy budgets. This ability to specify accuracy expectations exemplifies why-DP-provenance in action, as it provides valuable information about the usefulness and confidence intervals of query results. This feature ultimately enhances the user experience with DP systems. To achieve this user-centric approach, APEx made several vital contributions:

- A New Language for DP Queries: APEx designed a new, SQL-like query language tailored for DP tasks. This language simplifies query specification for analysts.

- Accuracy-Privacy Translation Framework: APEx developed a framework that automatically translates an analyst's desired accuracy level into an optimal DP mechanism. This framework eliminates the need for analysts to choose a mechanism themselves. APEx also compiled a comprehensive library of the latest DP mechanisms, ensuring the framework has the best one to achieve the desired accuracy-privacy trade-off.

In addition to these why-provenance advancements, APEx also introduced a novel data-dependent DP mechanism that leverages how-DP-provenance. This mechanism tracks the noise added during query processing to minimize privacy costs while still meeting the specified accuracy requirements. We will delve deeper into each of these contributions in the following sections.

Query Language with Accuracy Measures. APEx aims to support SQL-like declarative query languages with accuracy specifications. The syntax of the query language is defined in the following format.

$$\begin{array}{l} \text{BIN } D \text{ ON } f(\cdot) \text{ WHERE } W = \{\phi_1, \ldots, \phi_L\} \\ [\text{HAVING } f(\cdot) > c\,] \\ [\text{ORDER BY } f(\cdot) \text{ LIMIT } k\,] \\ \text{ERROR } \alpha \text{ CONFIDENCE } 1-\beta\,; \end{array}$$

The meaning of this query syntax is to map a database D into bins of rows $\{b_1, \ldots, b_L\}$ based on a workload of predicates $W = \{\phi_1, \ldots, \phi_L\}$, where each bin b_i contains rows that satisfy the predicate ϕ_i. Then it applies the aggregation function $f()$ (e.g., count and sum) over each bin b_i. The two lines in the brackets are optional. A *workload counting query* (WCQ) is simply the query when $f()$ is a counting function that returns the bin size without the two optional lines. A similar query with the HAVING clause is an *iceberg counting query* (ICQ) that returns a list of bin identifiers b_i for which $f(b_i) > c$, and a query with the ORDER BY ... LIMIT clause is called the *top-k counting query* (TCQ) that returns the k bins that have the largest values for $f(b_i)$.

For a query with a numerical output like WCQ, there are multiple accuracy semantics considered by the literature, such as mean square error (MSE) [131], relative error [131], and (α, β)-accuracy [39]. APEx considers (α, β)-accuracy for WCQ and extends it to non-numerical queries like ICQ and TCQ. In particular, we say a mechanism M satisfies (α, β)-WCQ accuracy, if with a high probability $1-\beta$, for each predicate $\phi \in W$, the absolute difference between its noisy answer returned by M and its true answer is bounded by α, i.e.,

$$\Pr[\max_{\phi \in W} |M_\phi(D) - c_\phi(D)| \leq \alpha] \geq 1 - \beta.$$

For a non-numerical query like ICQ, APEx has two parts for its accuracy requirement:

$$\Pr[|\{\phi \in M(D) \mid c_\phi(D) < c - \alpha\}| > 0] \leq \beta$$

$$\Pr[|\{\phi \in (W - M(D)) \mid c_\phi(D) > c + \alpha\}| > 0] \leq \beta$$

simultaneously. This first part corresponds to the type I error that wrongly labels predicates with a true count less than c as $> c$. The second part is for the type II error that labels the predicates with a true count greater than c as $< c$. The accuracy definition is satisfied if, with a high probability of $1-\beta$, all predicates with true counts greater than $c+\alpha$ or less than $c-\alpha$ are correctly labelled. APEx extends the same logic to the accuracy requirement for TCQ in a similar flavor. Note that both (α, β)-ICQ accuracy and (α, β)-TCQ accuracy definitions consider *symmetric* errors. Definitions and mechanisms that extend to *asymmetric* errors are discussed and proposed in the MIDE system for private decision-making [50].

Accuracy-Privacy Translation Framework. For each query and its accuracy specification, APEx automatically finds the best mechanism that satisfies the accuracy specification of the query with the least privacy budget. First, APEx prepares all the existing DP mechanisms and executes them in two phases: (i) privacy cost estimation, which simulates how much privacy budget would be needed if the mechanism were used, and (ii) running the algorithm, which actually runs the mechanism and returns the answer to the data analysts. Second, APEx stores all the relevant state-of-the-art DP mechanisms for each query type since the best one depends on the query and the data. For example, for WCQ, APEx provides two data-independent translation mechanisms. The first data-independent mechanism is the baseline *Laplace mechanism*, which analyzes the error bounds of Laplace noise and obtains directly a closed-form expression of bounds on the translated privacy budgets. The second mechanism, *strategy-based mechanism*, uses the matrix mechanism [81, 82] to analyze the overlapping parts of the workload W and reuse the noisy intermediate results for the overlapping part to translate the same accuracy target into, in many cases, a tighter bound on privacy budget. The strategy-based mechanism is also data-independent and can be generalized to all the query types.

Data Dependent Accuracy-Privacy Translation. APEx considers a data-dependent accuracy-privacy translation mechanism for ICQ, an example of how-DP-provenance. An ICQ involves comparing the bin sizes with the threshold value c. For example, the data-dependent Laplace mechanism adds noise to the true bin sizes and compares the noisy bin sizes with the threshold. For example, given an accuracy target $\alpha = 10, \beta = 0.00001$, APEx will translate this into budget $\frac{\ln(1/2\beta)}{\alpha} = 0.85$ for the Laplace mechanism. However, if the bin sizes are far away from the threshold, let's say $c_\phi(D) = 1000$ and the threshold is $c = 100$ where the difference is 90 times α, then it may be sufficient only to use $\frac{0.85}{90} \approx 0.01$ privacy budget. As APEx does not know the distance between the true bin sizes and the threshold, it proposes a *multi-poking mechanism* that starts with a small privacy budget and tests multiple times with increasing privacy budgets until it can determine the noisy answer would satisfy the accuracy target confidently. Rather than drawing independent Laplace noise in each iteration, this multi-poking mechanism stores the privacy budgets and the noise used in previous "pokings" and samples correlated noise each time [77]. This correlated noise allows the overall privacy loss to be bounded by the privacy budget of the last poking instead of the sum of the privacy budgets over all the poking steps.

4.1.2 Improvements and Limitations

APEx offers several advantages that make it easier to conduct private data analysis: 1) *usability improvement via why-DP-provenance*, where APEx empowers data analysts, even those without prior privacy expertise, to achieve high accuracy in tasks like entity resolution, as shown in its user studies; 2) privacy improvement via how-DP provenance, where the new multi-poking mechanism significantly reduces the privacy budget needed for specific workloads (4 out of 8 ICQ workloads in studies) compared to traditional approaches. While APEx offers significant benefits, it is essential to acknowledge some limitations: 1) *compatibility with accuracy bounds*, for not all data privacy algorithms have clearly defined accuracy limitations, which restricts APEx's ability to incorporate them into its framework; 2) *runtime overheads*, for APEx requires running all the relevant mechanisms and storing/tracing the correlated noise calibration in the multi-poking mechanism, though they are relatively small and are only needed at runtime.

4.2 Case Study: DProvDB for Multi-Analyst DP

Unlike prior systems, DProvDB enforces privacy constraints, not only on the queries and the input data but also on each analyst. This design aims to achieve a fair distribution of privacy budget among data analysts and a tight privacy control per data analyst. DProvDB also leverages multiple how-DP provenance techniques for its DP mechanisms, including tracking sensitivity and stability of queries like Chorus [66] and tracking noise to responses for different analysts, considering how these responses might be interrelated over time. In this case study, we will highlight the privacy constraints for why-DP-provenance and the new noise-tracking technique for how-DP-provenance.

4.2.1 Problem and Technical Brief

DProvDB considers the problem of building an online query processing system for multiple data analysts, who are regulated not to collude but may break the regulation and collude. These data analysts also have different trust/privilege levels when accessing the data; for example, internal analysts shall use more global privacy budgets than external data analysts. DP systems before DProvDB do not distinguish data analysts, and naively tracing each analyst's queries independent of others can waste the global budget — if collusion happens, the privacy loss across data analysts is upper bounded by $\sum \epsilon_i, \sum \delta_i$ while it is lower bounded by $\max \epsilon_i, \max \delta_i$, where ϵ_i, δ_i is the privacy budget spent on each data analyst.

Unlike the basic DP system described in Section 3.1, DProvDB might reject an analyst's query if answering it would exhaust either the analyst's individual privacy budget or the total budget shared by all analysts. This ensures fair and controlled use of privacy resources. To enforce these privacy constraints (why-DP-provenance), DProvDB utilizes two key components:

- Privacy Provenance Table: This table tracks past queries and the privacy budget spent on each. It allows DProvDB to monitor individual and overall budget consumption.
- Custom DP Mechanisms: DProvDB designs specialized DP mechanisms for this multi-analyst environment. These mechanisms consider budget limitations when determining whether to answer a query.

Furthermore, DProvDB tackles minimizing the overall privacy loss even in scenarios where analysts might collaborate (collusion). Here, DProvDB leverages a technique based on the additive Gaussian mechanism. This technique reuses previously generated noisy outputs (how-DP-provenance) to answer new queries from other analysts to achieve the lower bound for the privacy loss at collusion while still providing useful results.

Next, we will introduce the building block DP mechanism in DProvDB that achieves the lower privacy bound when all analysts collude for a simple query. We will then present how it is used for online query processing and the necessary provenance information for its deployment in DProvDB.

Building Block: Additive Gaussian Mechanism. Consider two analysts A_1 and A_2 send the same query q with two different privacy budgets (ϵ_1, δ) and (ϵ_2, δ), respectively (W.L.O.G, assuming $\epsilon_1 > \epsilon_2$). If responding to each analyst separately with an independent Gaussian mechanism (Definition 2.5), i.e., $q(D) + \eta_1$ for A_1 and $q(D) + \eta_2$ for A_2, where $\eta_1 \sim \mathcal{N}(0, \sigma_1^2 I)$ for (ϵ_1, δ)-DP and $\eta_2 \sim \mathcal{N}(0, \sigma_2^2 I)$ for (ϵ_2, δ)-DP, then the overall privacy loss if these two analysts collude will be $(\epsilon_1 + \epsilon_2, 2\delta)$.

The additive Gaussian mechanism first processes the noisy response to A_1 with the standard Gaussian mechanism, $q(D) + \eta_1$ from the above distribution. Then, it reuses η_1 in its response to A_2 by returning $q(D) + \eta_1 + \eta'$, where $\eta_1' \sim \mathcal{N}(0, \sigma_2^2 - \sigma_1^2)$. If A_1 and A_2 do not collude, the privacy loss to each one of them is (ϵ_1, δ)-DP and (ϵ_2, δ)-DP respectively. However, if they collude, the overall privacy loss is bounded by (ϵ_1, δ)-DP as the most accurate response they can come up with is the noisy response to A_1.

Note that the additive Gaussian mechanism described above only works for *the identical queries* when the *privacy budget of the first processed query is always greater than that of the future queries*. This limitation poses challenges in online database systems when 1) two analysts' queries *only overlaps* (i.e., not exactly the same), and 2) a query received at a *later timestamp* has a larger privacy budget compared to the processed historical query. To address these challenges, DProvDB devised the additive Gaussian mechanism by carefully selecting, maintaining, and updating a set of historical query answers to different data analysts and their respective privacy consumption over time.

Query Answering using Views/Synopses. To solve the first problem of overlapping queries, DProvDB does not directly apply the additive Gaussian mechanism to the queries from the data analysts. Instead, it creates materialized private views or synopses of the data using the additive Gaussian mechanism and post-processes queries on these synopses. These views are essentially histograms (or contingency tables for multiple columns). They capture the distribution of data for specific attributes and allow for processing queries that involve linear combinations of the data points (like finding averages or sums). Synopses are formed by adding different noises to the true answer of each view, and post-processing these noisy synopses does not consume an additional privacy budget. Hence, even if queries from analysts partially overlap or differ entirely, as long as they can be processed using the same view, DProvDB will update the corresponding synopses for this view with the additive Gaussian mechanism and use the updated synopses to answer the queries.

Incremental Synopses Maintenance. To tackle the second problem on dynamic budget, DProvDB maintains the noisy synopses *adaptively* based on incoming queries submitted to the system. DProvDB, in particular, has *two layers of synopses*: 1) a *global synopsis* per view, and 2) a *local synopsis* per view and per analyst. The local synopsis is always generated from the global synopsis using the additive Gaussian mechanism, and the analyst's queries are always processed on their corresponding local synopses (viz., post-processing). Therefore, privacy loss across data analysts is always bounded by the privacy budget used for generating global synopses.

To answer queries with a higher privacy budget (i.e., the analyst wants the query answer to be more accurate) while reusing existing synopses, DProvDB updates the global synopses using the following approach. When the global DP synopsis V^{ϵ} does not provide enough accuracy to handle a local synopsis request at privacy budget ϵ_t, DProvDB spends additional privacy budget $\Delta\epsilon$ to update the global DP synopsis to $V^{\epsilon+\Delta\epsilon}$, where $\Delta\epsilon = \epsilon_t - \epsilon$. Here, DProvDB uses the standard Gaussian mechanism, which generates an intermediate DP synopsis $V^{\Delta\epsilon}$ with a budget $\Delta\epsilon$, and then combines the previous synopses with this intermediate synopsis into an updated one. The key insight of the combination is to properly involve the fresh noisy synopses by assigning each synopsis with a weight proportional to the inverse of its noise variance, which gives the smallest expected square error based on UMVUE [71, 105]. That is, for the t-th release, we combine these two synopses $V^{\epsilon_t} = (1 - w_t)V^{\epsilon_{t-1}} + w_t V^{\Delta\epsilon}$. The resulted expected square error for V^{ϵ_t} is $v_t = (1 - w_t)^2 v_{t-1} + w_t^2 v_{\Delta}$, where v_{t-1} is the noise variance of view $V^{\epsilon_{t-1}}$, and v_{Δ} is derived from $V^{\Delta\epsilon}$. The error is minimized at $w_t = \frac{v_{t-1}}{v_{\Delta}+v_{t-1}}$.

Privacy Provenance Table. Besides maintaining the global and local synopses, DProvDB keeps a privacy provenance table to manage the privacy budgets. The privacy provenance table $\mathcal{P}$ consists of (i) a provenance matrix P that tracks the privacy loss of a view in $\mathcal{V}$ to each data analyst in $\mathcal{A}$, where each entry of the matrix $P[A_i, V_j]$ records the current cumulative privacy loss $S^{A_i}_{V_j}$, on view V_j to analyst A_i; (ii) a set of row/column/table constraints, Ψ: a <u>row constraint</u> for i-th row of P, denoted by ψ_{A_i}, refers to the allowed maximum privacy loss to a data analyst $A_i \in \mathcal{A}$ (according to his/her privilege level); a <u>column constraint</u> for the j-th column, denoted by ψ_{V_j} refers to as the allowed maximum privacy loss to a specific view V_j; the <u>table constraint</u> over P, denoted by ψ_P, specifies the overall privacy loss allowed for the protected database. Due to the privacy constraints imposed by the privacy provenance table, queries can be rejected when the cumulative privacy cost exceeds the constraints. The overall privacy guarantee of the system is then implied by the three levels of privacy constraints over the provenance table. Given the privacy provenance table and its constraint specifications, $\Psi = \{\psi_{A_i} | A_i \in \mathcal{A}\} \cup \{\psi_{V_j} | V_j \in \mathcal{V}\} \cup \{\psi_P\}$, DProvDB ensures $[\ldots, (A_i, \psi_{A_i}, \delta), \ldots]$-multi-analyst-DP; it also ensures $\min(\psi_{V_j}, \psi_P)$-DP for view $V_j \in \mathcal{V}$ and overall ψ_P-DP if all the data analysts collude.

4.2.2 Improvements and Limitations

DProvDB is built as a middleware or a multi-analyst interface that works on top of the existing Chorus system [66]. This allows DProvDB to leverage Chorus's functionalities while adding its own capabilities. Experiments of DProvDB are tested over the Adult census dataset [38] and the TPC-H synthetic dataset [20] with two types of workloads, one of which consists of randomized range queries over random attributes while the other simulates traversing a decomposition tree of the domain of selected attributes. With more than one data analyst in the experimental setup, empirical results show that DProvDB dominates existing DP query processing systems by answering 2.5x-1000x more queries given the same privacy budget.

One current limitation of DProvDB is the overhead associated with storing, querying, and updating the privacy provenance information (privacy provenance table). Future work will focus on optimizing these operations for better efficiency. Interestingly, the way DProvDB updates synopses based on analyst queries is similar to the problem of *incremental view maintenance* from the field of data provenance. While the current algorithm in DProvDB does not modify the actual queries, there is potential to explore how incremental view maintenance techniques could inspire new and more efficient algorithms for private data management. There are also several interesting future directions related to privacy provenance for multi-analyst DP. For analyst provenance tracking, research questions and works may be spawned by a deeper intertwinement between privacy provenance and access/leakage control [98] or focus on a more expressive model for privacy provenance. For example, an analyst may temporarily delegate his/her privacy privilege to other analysts.

4.3 Case Study: User-Level Adaptive Block Composition in Sage and Cohere

Systems like Sage [80] and Cohere [78] aim to build a DP system that can continuously run with a finite global budget. To achieve this, Sage and Cohere, different from the basic DP system, enable more fine-grained privacy accounting (i.e., where-DP-provenance) at the level of subsets (i.e., blocks) of the data and replace the retired data blocks with new data. In addition, Cohere achieves user-level DP (privacy resolutions in where-DP-provenance) and enables budget allocation optimization over a batch of applications/queries (i.e., features how-DP-provenance).

4.3.1 Problem and Technical Brief

Sage and Cohere study the approaches to building DP systems that can continuously run with a finite global budget. They explore the heterogeneous input data streams and the parallel/block composition techniques that account for privacy loss over disjoint subsets of data.

Block Composition in Sage [80]. Block composition is an extension of parallel composition to the streaming data model. To apply block composition, at each timestamp, the system will create a new data block with disjoint sets of data from the previous blocks and also maintain the state of the block with a privacy filter [108]. Note that the creation/split of the block is based on some publicly known specifications while the data blocks after splitting is remaining secret. The specifications are criteria of how data is split over public domain values of certain attributes, e.g., timestamp, userID, geography. At query time, the system allows to run DP queries adaptively over the overlapping subsets of the blocks created so far. The privacy accounting is performed by updating the privacy filter per block, which is intuitively enforcing adaptive sequential composition (or other tighter composition bounds [79]) within a block and parallel composition across blocks (if a query is answered using multiple blocks)[3]. Sage applies block composition over time splits and hence guarantees event-level DP. The subsequent work, Cohere [78], extends the methodology of block composition and applies to user-level DP.

Partitioning Attributes and User Rotation in Cohere [78]. Cohere creates new blocks based on split over both userIDs and a (set of) given attribute(s). That is, each block generated contains a new batch of users that never appear in the previous blocks and a value in the selected partitioning attributes. For example, data block 1 contains users 1-3 all with region A, and data block 2 has users 4-6 all with region B, and block 3 consists of users 7-9 with region A, etc. Cohere then applies block composition over the user data blocks and retires users with replacement of new users to keep the system continuously running. Cohere also adopts a user rotation mechanism to prevent users from retiring too quickly from certain subpopulations (in terms of some values of the partitioning attributes), which reduces biases in answering queries or running applications. The approach is based on sliding windows (or, in essence, the least recently used strategy) so that it is independent of the user attributes. In particular, at each timestamp, the newly joined users are randomly partitioned/assigned into groups, and the group that was active the longest will be retired (tentatively if the budget over this group is not depleted), and a new group will be activated. The budget spent by queries at this timestamp will be capped with $1/K$ of the global budget where K is the number of active groups.

[3] By using privacy filter, Sage can support strong composition [42, 69] with block composition.

Formalizing Optimization Problem in Cohere [78]. Cohere further uses the tracked budget allocation history per block (or where-DP-provenance) to develop an optimization problem, as a variant of the multidimensional knapsack problem, for query answering. Each query R_i in the Cohere system is annotated with a propositional formula Φ_i over the partitioning attributes, a privacy budget $\mathbf{C}_i \in \mathbb{R}^{|\mathcal{A}|}_{\geq 0}$, and a weight $W_i \in \mathbb{N}$. The goal of Cohere's query processing is to batch all the queries $\mathcal{R}$ received at the current timestamp, and find an optimal (in terms of the weights) set of queries to answer while subject to privacy constraints due to partitioning attributes. To formulate the optimization problem, Cohere introduces decision variables $y_i \in \{0, 1\}$ for $i \in \mathcal{R}$, where $y_i = 1$ means the request R_i is accepted, and $y_i = 0$ means the request has been rejected. The privacy constraints are defined over blocks $\mathcal{S}$. A block $S_j \in \mathcal{S}$ is denoted by $(groupid, \Psi_j, \mathbf{B}_j)$, where Ψ_j is a propositional formula over the partitioning attributes and $\mathbf{B}_j \in \mathbb{R}^{A}_{\geq 0}$ is the remaining budget for this block. Given the demand for a block, written as $d_{ij} = \mathbf{C}_i$ if $\Phi_i \wedge \Psi_j$ is satisfiable, and 0 otherwise, the optimization problem is formalized as $\max \sum_{i \in \mathcal{R}} y_i \cdot W_i,$ s.t. $\sum_{i \in \mathcal{R}} d_{ij} y_i \leq B_j,\ [\forall j \in \mathcal{S}]$. Cohere generalizes the problem into an integer linear programming (ILP) problem and solves it with an ILP solver. Cohere also shows an optimization technique to reduce the dimensionalities to scale the solving process.

4.3.2 Improvements and Limitations

Sage and Cohere report the benefit of enabling block composition and recording per block budget consumption (viz., where-DP-provenance) through extensive experimental evaluations. Sage shows that with block composition, the system can train a machine learning model with a lower mean squared error (MSE) that cannot be achieved by sequential composition (Δ=0.0002); to achieve the same MSE, block composition requires much less data than sequential composition (10x-100x less in certain cases). On the other hand, Cohere is compared with PrivateKube [84], a privacy budget scheduler built based on Sage. In an end-to-end comparison with PrivateKube, Cohere shows an improvement in handling 1.5x-2.0x more queries and a 6.4x–28x better utility due to the partitioning attributes approach and the more fine-grained privacy analysis. All these experimental results on Sage and Cohere validate the effectiveness of enabling where-DP-provenance in a DP system.

However, Cohere cannot be run in real-time systems since maintaining the block composition and solving the optimization problem requires, on average, 48 minutes and around 1.3 GB of memory. Empirical results also observe an increasing runtime when using the partitioning attribute approach. Another limitation of using block composition or partitioning attributes with where-DP-provenance is that they assume the criteria for creating the partitioning is publicly known; otherwise, it will cause problems in the privacy analysis. Removing the assumption may be a future direction to explore for where-DP-provenance.

5 Discussion and Open Questions

This section discusses the fine-grained provenance for DP systems related to traditional data provenance literature and the development of DP with their respective open questions.

5.1 Relationship with Data Provenance

The well-established field of data provenance offers valuable insights for the future development of DP provenance. In this work, we explored why, how, and where provenance for DP systems. To delve deeper, let's discuss two additional areas for consideration.

Representations of Privacy Provenance. In traditional databases, provenance information can be represented using either an eager or a lazy approach. The eager approach [15] attaches extra metadata (annotations) to queries and propagates it to the results, e.g., based on the provenance semirings [54] or calculation of Shapley value [85, 86]. While this allows for direct retrieval of provenance information, it can incur performance overhead and require additional storage for the metadata. The lazy approach [18] relies on properties of specific transformations to identify the source data behind the output without annotations. This method has lower overhead but limited applicability. In the context of DP provenance, we have primarily focused on leveraging and storing additional metadata, similar to the eager approach. It would be interesting to explore the feasibility of a lazy approach for DP provenance. This could involve developing mechanisms to answer why-, how-, and where-provenance queries without needing constant metadata storage and updates.

Scalable Privacy Provenance Tracking. Research in database provenance has addressed the challenge of scalability in managing provenance information [49, 112, 16, 106]. These methods aim to reduce the cost of tracking provenance by either minimizing the amount of extra metadata required or employing compression techniques to approximate provenance. For DP provenance, the level of granularity (detail) directly impacts the amount of data that needs to be tracked. Finer-grained provenance necessitates tracking more data. There are two key areas for further exploration. First, we would like to have a better understanding of the trade-off between privacy granularity and the associated storage and processing overhead. For instance, DProvDB can use a finer-grained caching mechanism like CacheDP [87] to further save privacy budget per query, but maintaining and updating such cache structures for all analysts can be very expensive. This will guide future research efforts. Second, developing techniques to reduce the cost of privacy provenance tracking is a promising research direction. Existing systems, like DProvDB, which tracks analyst provenance at the view level for efficiency, offer valuable insights. Future work could explore compression and approximation techniques specifically tailored for efficient privacy provenance management.

5.2 Relationship with the Development of DP

Differential privacy has been around since 2006 and has evolved significantly. Today, we have a vast array of DP algorithms for various uses, different privacy definitions for various scenarios, and even prototype programming tools and systems. This article focuses on DP systems that leverage provenance techniques to improve usability or performance. The effectiveness of these provenance techniques is directly related to the development of DP algorithms and definitions. Let's explore some key areas for further exploration.

Optimal Algorithm Design. Accuracy-first mechanisms are less understood compared to privacy-first mechanisms. For accuracy-first mechanisms, how queries translate into privacy guarantees can vary depending on the specific queries, how accuracy is measured by the system, and even the data (e.g., joining tables or measuring relative error). Existing research for privacy-first mechanisms has produced optimal solutions for specific queries like joins [32, 30, 29, 131]. However, there is a gap in understanding how to achieve optimal results for accuracy-first mechanisms, particularly those that depend on the data. Closing this gap is crucial for developing user-friendly DP systems that leverage why-DP-provenance.

Recent advancements in DP mechanisms [87, 48, 100, 126] involve using correlated noise drawn from different points over time. This approach can lead to tighter privacy analysis or improved utility for the results. However, effectively and securely maintaining these noise sequences is critical for successful deployment. This necessitates the development of systematic how-DP-provenance techniques in the future.

Limited DP algorithms have been developed specifically for growing data models, and they often overlook how data evolves over time (e.g., their temporal properties). In machine learning, for instance, data patterns and learned models can change over time (i.e., the concept drift). Factoring in concept drift will likely require even finer-grained how/where-DP-provenance tracking for effective solutions.

Mixing DP Variants. DP provides some degree of freedom to allow system designers to "composite" privacy guarantees. However, a key challenge arises if one part of the data is released with DP while others are queried and processed with other privacy notions, such as OSDP [76], attribute privacy [140], pufferfish privacy [72], etc., each with its own strengths and use cases. While combining these use cases into a unified system might be desirable, a significant question remains: how do we account for the total privacy loss when mixing different privacy-preserving techniques? Recent research in encrypted databases [139] explores similar challenges in reasoning about security when combining multiple encryption techniques. This offers valuable insights for the DP domain. An interesting future direction would be to develop techniques specifically for mixing different privacy definitions and calculating the resulting privacy loss. One possibility is to leverage existing DP auditing techniques [101, 62, 91]. These techniques can help us establish a lower bound for the overall privacy guarantee, even when combining different privacy-preserving methods.

Removing Trusted Curators in DP. Our discussion of privacy provenance has so far focused on centralized DP systems, which rely on a trusted curator to oversee the entire process. However, this centralized approach may not be practical in all real-world scenarios. Removing the need for a trusted party is an active area of research with several promising directions. First, local DP empowers data owners to add noise to their own data before it is used in queries. While this offers greater privacy control, it can lead to lower accuracy than centralized DP. Existing research has explored using anonymous shufflers [11, 52] to improve utility in local DP settings. An interesting future direction would be to investigate how other privacy provenance information, besides shuffling, can be leveraged to enhance utility in local DP systems. Second, combining DP with cryptography [122, 109] offers another approach to reduce the noise needed for the local or federated settings [13, 7]. Third, enabling a DP system with trusted hardware, e.g., SGX [94], can simulate the trusted curator in the untrusted settings. However, as shown in recent work [63], the last two approaches have to maintain and track a significant amount of additional metadata about the state of the running environment. Without this metadata tracking, the system remains vulnerable. Privacy provenance can be crucial in future work on these decentralized privacy-preserving techniques. Providing a systematic view of data usage and privacy guarantees can help address the challenges associated with removing the need for a trusted central authority in DP systems.

6 Conclusion

This article explores how provenance techniques can empower differential privacy (DP) systems. We introduce a novel taxonomy for three crucial DP provenance types (why-provenance, how-provenance, and where-provenance). We then unpack existing techniques for each type, leveraging case studies to illuminate their advantages and drawbacks. Finally, we establish the link between fine-grained DP provenance and traditional data provenance, investigating how both can propel advancements in DP across various domains. This work is the first attempt to bridge these two critical areas. We hope it offers a unique perspective and paves the way for further research in this exciting direction.

References

[1] J. M. Abowd. The us census bureau adopts differential privacy. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pages 2867–2867, 2018.

[2] C. Abuah, A. Silence, D. Darais, and J. P. Near. Dduo: General-purpose dynamic analysis for differential privacy. In 2021 IEEE 34th Computer Security Foundations Symposium (CSF), pages 1–15. IEEE, 2021.

[3] C. Abuah, D. Darais, and J. P. Near. Solo: A lightweight static analysis for differential privacy. Proc. ACM Program. Lang., 6(OOPSLA2), oct 2022. doi: 10.1145/3563313. URL https://doi.org/10.1145/3563313.

[4] Amazon Inc. AWS Clean Rooms differential privacy. https://aws.amazon.com/clean-rooms/differential-privacy/. Accessed: 2024-05-31.

[5] K. Amin, J. Gillenwater, M. Joseph, A. Kulesza, and S. Vassilvitskii. Plume: Differential privacy at scale. CoRR, abs/2201.11603, 2022. URL https://arxiv.org/abs/2201.11603.

[6] B. Balle and Y. Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In J. G. Dy and A. Krause, editors, Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, volume 80 of Proceedings of Machine Learning Research, pages 403–412. PMLR, 2018. URL http://proceedings.mlr.press/v80/balle18a.html.

[7] E. Bao, Y. Zhu, X. Xiao, Y. Yang, B. C. Ooi, B. H. M. Tan, and K. M. M. Aung. Skellam mixture mechanism: a novel approach to federated learning with differential privacy. Proc. VLDB Endow., 15(11): 2348–2360, 2022. URL https://www.vldb.org/pvldb/vol15/p2348-bao.pdf.

[8] G. Barthe, R. Chadha, V. Jagannath, A. P. Sistla, and M. Viswanathan. Deciding differential privacy for programs with finite inputs and outputs. In Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science, pages 141–154, 2020.

[9] B. Bichsel, T. Gehr, D. Drachsler-Cohen, P. Tsankov, and M. Vechev. Dp-finder: Finding differential privacy violations by sampling and optimization. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pages 508–524, 2018.

[10] B. Bichsel, S. Steffen, I. Bogunovic, and M. Vechev. Dp-sniper: Black-box discovery of differential privacy violations using classifiers. In 2021 IEEE Symposium on Security and Privacy (SP), pages 391–409. IEEE, 2021.

[11] A. Bittau, Ú. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnes, and B. Seefeld. Prochlo: Strong privacy for analytics in the crowd. In Proceedings of the 26th symposium on operating systems principles, pages 441–459, 2017.

[12] D. M. Bittner, A. E. Brito, M. Ghassemi, S. Rane, A. D. Sarwate, and R. N. Wright. Understanding privacy-utility tradeoffs in differentially private online active learning. Journal of Privacy and Confidentiality, 10 (2), 2020.

[13] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 1175–1191, 2017.

[14] P. Buneman, S. Khanna, and W. C. Tan. Why and where: A characterization of data provenance. In J. V. den Bussche and V. Vianu, editors, Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings, volume 1973 of Lecture Notes in Computer Science, pages 316–330. Springer, 2001. doi: 10.1007/3-540-44503-X_20. URL https://doi.org/10.1007/3-540-44503-X_20.

[15] P. Buneman, S. Khanna, and W. C. Tan. On propagation of deletions and annotations through views. In L. Popa, S. Abiteboul, and P. G. Kolaitis, editors, Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA, pages 150–158. ACM, 2002. doi: 10.1145/543613.543633. URL https://doi.org/10.1145/543613.543633.

[16] A. P. Chapman, H. V. Jagadish, and P. Ramanan. Efficient provenance storage. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pages 993–1006, 2008.

[17] K. Chatzikokolakis, M. E. Andrés, N. E. Bordenabe, and C. Palamidessi. Broadening the scope of differential privacy using metrics. In E. D. Cristofaro and M. K. Wright, editors, Privacy Enhancing Technologies - 13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013.

Proceedings, volume 7981 of Lecture Notes in Computer Science, pages 82–102. Springer, 2013. doi: 10.1007/978-3-642-39077-7_5. URL `https://doi.org/10.1007/978-3-642-39077-7_5`.

[18] J. Cheney, L. Chiticariu, and W. C. Tan. Provenance in databases: Why, how, and where. Found. Trends Databases, 1(4):379–474, 2009. doi: 10.1561/1900000006. URL `https://doi.org/10.1561/1900000006`.

[19] E. Cohen, X. Lyu, J. Nelson, T. Sarlós, and U. Stemmer. Optimal differentially private learning of thresholds and quasi-concave optimization. In Proceedings of the 55th Annual ACM Symposium on Theory of Computing, pages 472–482, 2023.

[20] T. T. P. P. Council. The tpc benchmark h (tpc-h)., 2008. URL `https://www.tpc.org/tpch/`.

[21] C. Covington, X. He, J. Honaker, and G. Kamath. Unbiased statistical estimation and valid confidence intervals under differential privacy. Statistica Sinica, to appear.

[22] R. Cummings, S. Krehbiel, K. A. Lai, and U. Tantipongpipat. Differential privacy for growing databases. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS'18, page 8878–8887, Red Hook, NY, USA, 2018. Curran Associates Inc.

[23] R. Cummings, D. Desfontaines, D. Evans, R. Geambasu, Y. Huang, M. Jagielski, P. Kairouz, G. Kamath, S. Oh, O. Ohrimenko, N. Papernot, R. Rogers, M. Shen, S. Song, W. Su, A. Terzis, A. Thakurta, S. Vassilvitskii, Y.-X. Wang, L. Xiong, S. Yekhanin, D. Yu, H. Zhang, and W. Zhang. Advancing Differential Privacy: Where We Are Now and Future Directions for Real-World Deployment. Harvard Data Science Review, 6(1), jan 16 2024. https://hdsr.mitpress.mit.edu/pub/sl9we8gh.

[24] S. B. Davidson, S. Khanna, T. Milo, D. Panigrahi, and S. Roy. Provenance views for module privacy. In M. Lenzerini and T. Schwentick, editors, Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece, pages 175–186. ACM, 2011. doi: 10.1145/1989284.1989305. URL `https://doi.org/10.1145/1989284.1989305`.

[25] S. B. Davidson, S. Khanna, S. Roy, J. Stoyanovich, V. Tannen, and Y. Chen. On provenance and privacy. In T. Milo, editor, Database Theory - ICDT 2011, 14th International Conference, Uppsala, Sweden, March 21-24, 2011, Proceedings, pages 3–10. ACM, 2011. doi: 10.1145/1938551.1938554. URL `https://doi.org/10.1145/1938551.1938554`.

[26] D. Deutch, A. Frankenthal, A. Gilad, and Y. Moskovitch. On optimizing the trade-off between privacy and utility in data provenance. In G. Li, Z. Li, S. Idreos, and D. Srivastava, editors, SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021, pages 379–391. ACM, 2021. doi: 10.1145/3448016.3452835. URL `https://doi.org/10.1145/3448016.3452835`.

[27] B. Ding, J. Kulkarni, and S. Yekhanin. Collecting telemetry data privately. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 3571–3580, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/253614bbac999b38b5b60cae531c4969-Abstract.html`.

[28] Z. Ding, Y. Wang, G. Wang, D. Zhang, and D. Kifer. Detecting violations of differential privacy. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pages 475–489, 2018.

[29] W. Dong and K. Yi. Residual sensitivity for differentially private multi-way joins. In G. Li, Z. Li, S. Idreos, and D. Srivastava, editors, SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021, pages 432–444. ACM, 2021. doi: 10.1145/3448016.3452813. URL https://doi.org/10.1145/3448016.3452813.

[30] W. Dong and K. Yi. A nearly instance-optimal differentially private mechanism for conjunctive queries. In L. Libkin and P. Barceló, editors, PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022, pages 213–225. ACM, 2022. doi: 10.1145/3517804.3524143. URL https://doi.org/10.1145/3517804.3524143.

[31] W. Dong and K. Yi. Universal private estimators. In Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, pages 195–206, 2023.

[32] W. Dong, J. Fang, K. Yi, Y. Tao, and A. Machanavajjhala. R2T: instance-optimal truncation for differentially private query evaluation with foreign keys. In Z. G. Ives, A. Bonifati, and A. E. Abbadi, editors, SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022, pages 759–772. ACM, 2022. doi: 10.1145/3514221.3517844. URL https://doi.org/10.1145/3514221.3517844.

[33] W. Dong, Q. Luo, and K. Yi. Continual observation under user-level differential privacy. In 44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023, pages 2190–2207. IEEE, 2023. doi: 10.1109/SP46215.2023.10179466. URL https://doi.org/10.1109/SP46215.2023.10179466.

[34] W. Dong, D. Sun, and K. Yi. Better than composition: How to answer multiple relational queries under differential privacy. Proc. ACM Manag. Data, 1(2):123:1–123:26, 2023. doi: 10.1145/3589268. URL https://doi.org/10.1145/3589268.

[35] W. Dong, Z. Chen, Q. Luo, E. Shi, and K. Yi. Continual observation of joins under differential privacy. Proceedings of the ACM on Management of Data, 2(3):1–27, 2024.

[36] J. Drechsler, I. Globus-Harris, A. Mcmillan, J. Sarathy, and A. Smith. Nonparametric differentially private confidence intervals for the median. Journal of Survey Statistics and Methodology, 10(3):804–829, 2022.

[37] W. Du, C. Foot, M. Moniot, A. Bray, and A. Groce. Differentially private confidence intervals. arXiv preprint arXiv:2001.02285, 2020.

[38] D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[39] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci., 9(3-4):211–407, 2014. doi: 10.1561/0400000042. URL https://doi.org/10.1561/0400000042.

[40] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith. Calibrating noise to sensitivity in private data analysis. In S. Halevi and T. Rabin, editors, Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings, volume 3876 of Lecture Notes in Computer Science, pages 265–284. Springer, 2006. doi: 10.1007/11681878_14. URL https://doi.org/10.1007/11681878_14.

[41] C. Dwork, M. Naor, T. Pitassi, G. N. Rothblum, and S. Yekhanin. Pan-private streaming algorithms. In A. C. Yao, editor, Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January

5-7, 2010. Proceedings, pages 66–80. Tsinghua University Press, 2010. URL http://conference.iiis.tsinghua.edu.cn/ICS2010/content/papers/6.html.

[42] C. Dwork, G. N. Rothblum, and S. P. Vadhan. Boosting and differential privacy. In 51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA, pages 51–60. IEEE Computer Society, 2010. doi: 10.1109/FOCS.2010.12. URL https://doi.org/10.1109/FOCS.2010.12.

[43] C. Dwork, M. Naor, and S. P. Vadhan. The privacy of the analyst and the power of the state. In 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012, pages 400–409. IEEE Computer Society, 2012. doi: 10.1109/FOCS.2012.87. URL https://doi.org/10.1109/FOCS.2012.87.

[44] H. Ebadi, D. Sands, and G. Schneider. Differential privacy: Now it's getting personal. In S. K. Rajamani and D. Walker, editors, Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015, pages 69–81. ACM, 2015. doi: 10.1145/2676726.2677005. URL https://doi.org/10.1145/2676726.2677005.

[45] J. Fang, W. Dong, and K. Yi. Shifted inverse: A general mechanism for monotonic functions under user differential privacy. In H. Yin, A. Stavrou, C. Cremers, and E. Shi, editors, Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022, pages 1009–1022. ACM, 2022. doi: 10.1145/3548606.3560567. URL https://doi.org/10.1145/3548606.3560567.

[46] M. Gaboardi, J. Honaker, G. King, J. Murtagh, K. Nissim, J. Ullman, and S. Vadhan. Psi ({\Psi}): a private data sharing interface. arXiv preprint arXiv:1609.04340, 2016.

[47] A. Gadotti, F. Houssiau, M. S. M. S. Annamalai, and Y. de Montjoye. Pool inference attacks on local differential privacy: Quantifying the privacy guarantees of apple's count mean sketch in practice. In K. R. B. Butler and K. Thomas, editors, 31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022, pages 501–518. USENIX Association, 2022. URL https://www.usenix.org/conference/usenixsecurity22/presentation/gadotti.

[48] C. Ge, X. He, I. F. Ilyas, and A. Machanavajjhala. Apex: Accuracy-aware differentially private data exploration. In P. A. Boncz, S. Manegold, A. Ailamaki, A. Deshpande, and T. Kraska, editors, Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019, pages 177–194. ACM, 2019. doi: 10.1145/3299869.3300092. URL https://doi.org/10.1145/3299869.3300092.

[49] F. Geerts, A. Kementsietsidis, and D. Milano. Mondrian: Annotating and querying databases through colors and blocks. In 22nd International Conference on Data Engineering (ICDE'06), pages 82–82. IEEE, 2006.

[50] S. Ghayyur, D. Ghosh, X. He, and S. Mehrotra. MIDE: accuracy aware minimally invasive data exploration for decision support. Proc. VLDB Endow., 15(11):2653–2665, 2022. URL https://www.vldb.org/pvldb/vol15/p2653-ghayyur.pdf.

[51] B. Ghazi, R. Kumar, P. Manurangsi, and T. Steinke. Algorithms with more granular differential privacy guarantees. In Y. T. Kalai, editor, 14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA, volume 251 of LIPIcs, pages 54:1–54:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi: 10.4230/LIPICS.ITCS.2023.54. URL https://doi.org/10.4230/LIPIcs.ITCS.2023.54.

[52] A. M. Girgis, D. Data, S. Diggavi, A. T. Suresh, and P. Kairouz. On the renyi differential privacy of the shuffle model. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pages 2321–2341, 2021.

[53] Google Inc. Use differential privacy – Big Query Documentations. `https://cloud.google.com/bigquery/docs/differential-privacy`. Accessed: 2024-05-31.

[54] T. J. Green and V. Tannen. The semiring framework for database provenance. In E. Sallinger, J. V. den Bussche, and F. Geerts, editors, Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017, pages 93–99. ACM, 2017. doi: 10.1145/3034786.3056125. URL `https://doi.org/10.1145/3034786.3056125`.

[55] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In L. Libkin, editor, Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China, pages 31–40. ACM, 2007. doi: 10.1145/1265530.1265535. URL `https://doi.org/10.1145/1265530.1265535`.

[56] M. Hardt and G. N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In 51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA, pages 61–70. IEEE Computer Society, 2010. doi: 10.1109/FOCS.2010.85. URL `https://doi.org/10.1109/FOCS.2010.85`.

[57] M. B. Hawes. Implementing differential privacy: Seven lessons from the 2020 united states census. Harvard Data Science Review, 2(2):4, 2020.

[58] M. Hay, A. Machanavajjhala, G. Miklau, Y. Chen, D. Zhang, and G. Bissias. Exploring privacy-accuracy tradeoffs using dpcomp. In Proceedings of the 2016 International Conference on Management of Data, pages 2101–2104, 2016.

[59] X. He, A. Machanavajjhala, and B. Ding. Blowfish privacy: tuning privacy-utility trade-offs using policies. In C. E. Dyreson, F. Li, and M. T. Özsu, editors, International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014, pages 1447–1458. ACM, 2014. doi: 10.1145/2588555.2588581. URL `https://doi.org/10.1145/2588555.2588581`.

[60] F. Houssiau, L. Rocher, and Y.-A. de Montjoye. On the difficulty of achieving differential privacy in practice: user-level guarantees in aggregate location data. Nature communications, 13(1):29, 2022.

[61] J. Hsu, A. Roth, and J. R. Ullman. Differential privacy for the analyst via private equilibrium computation. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013, pages 341–350. ACM, 2013. doi: 10.1145/2488608.2488651. URL `https://doi.org/10.1145/2488608.2488651`.

[62] M. Jagielski, J. R. Ullman, and A. Oprea. Auditing differentially private machine learning: How private is private sgd? In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/fc4ddc15f9f4b4b06ef7844d6bb53abf-Abstract.html`.

[63] J. Jin, C. Chuengsatiansup, T. Murray, B. I. P. Rubinstein, Y. Yarom, and O. Ohrimenko. Elephants do not forget: Differential privacy with state continuity for privacy budget. CoRR, abs/2401.17628, 2024. doi: 10.48550/ARXIV.2401.17628. URL `https://doi.org/10.48550/arXiv.2401.17628`.

[64] M. F. S. John, G. Denker, P. Laud, K. Martiny, A. Pankova, and D. Pavlovic. Decision support for sharing data using differential privacy. In 2021 IEEE Symposium on Visualization for Cyber Security (VizSec), pages 26–35. IEEE, 2021.

[65] N. M. Johnson, J. P. Near, and D. Song. Towards practical differential privacy for SQL queries. Proc. VLDB Endow., 11(5):526–539, 2018. doi: 10.1145/3187009.3177733. URL `http://www.vldb.org/pvldb/vol11/p526-johnson.pdf`.

[66] N. M. Johnson, J. P. Near, J. M. Hellerstein, and D. Song. Chorus: a programming framework for building scalable differential privacy mechanisms. In IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020, pages 535–551. IEEE, 2020. doi: 10.1109/EuroSP48549.2020.00041. URL `https://doi.org/10.1109/EuroSP48549.2020.00041`.

[67] Z. Jorgensen, T. Yu, and G. Cormode. Conservative or liberal? personalized differential privacy. In J. Gehrke, W. Lehner, K. Shim, S. K. Cha, and G. M. Lohman, editors, 31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015, pages 1023–1034. IEEE Computer Society, 2015. doi: 10.1109/ICDE.2015.7113353. URL `https://doi.org/10.1109/ICDE.2015.7113353`.

[68] P. Kairouz, S. Oh, and P. Viswanath. The composition theorem for differential privacy. In F. Bach and D. Blei, editors, Proceedings of the 32nd International Conference on Machine Learning, volume 37 of Proceedings of Machine Learning Research, pages 1376–1385, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/kairouz15.html`.

[69] P. Kairouz, S. Oh, and P. Viswanath. The composition theorem for differential privacy. IEEE Trans. Inf. Theory, 63(6):4037–4049, 2017. doi: 10.1109/TIT.2017.2685505. URL `https://doi.org/10.1109/TIT.2017.2685505`.

[70] G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias. Differentially private event sequences over infinite streams. Proceedings of the VLDB Endowment, 7(12), 2014.

[71] J. Kiefer. On minimum variance estimators. The Annals of Mathematical Statistics, 23(4):627–629, 1952.

[72] D. Kifer and A. Machanavajjhala. Pufferfish: A framework for mathematical privacy definitions. ACM Transactions on Database Systems (TODS), 39(1):1–36, 2014.

[73] K. Knopf. Framework for differentially private data analysis with multiple accuracy requirements. In G. Li, Z. Li, S. Idreos, and D. Srivastava, editors, SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021, pages 2890–2892. ACM, 2021. doi: 10.1145/3448016.3450587. URL `https://doi.org/10.1145/3448016.3450587`.

[74] K. Kostopoulou, P. Tholoniat, A. Cidon, R. Geambasu, and M. Lécuyer. Turbo: Effective caching in differentially-private databases. In J. Flinn, M. I. Seltzer, P. Druschel, A. Kaufmann, and J. Mace, editors, Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023, pages 579–594. ACM, 2023. doi: 10.1145/3600006.3613174. URL `https://doi.org/10.1145/3600006.3613174`.

[75] I. Kotsogiannis, Y. Tao, X. He, M. Fanaeepour, A. Machanavajjhala, M. Hay, and G. Miklau. Privatesql: A differentially private SQL query engine. Proc. VLDB Endow., 12(11):1371–1384, 2019. doi: 10.14778/3342263.3342274. URL `http://www.vldb.org/pvldb/vol12/p1371-kotsogiannis.pdf`.

[76] I. Kotsogiannis, S. Doudalis, S. Haney, A. Machanavajjhala, and S. Mehrotra. One-sided differential privacy. In 36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020, pages 493–504. IEEE, 2020. doi: 10.1109/ICDE48307.2020.00049. URL https://doi.org/10.1109/ICDE48307.2020.00049.

[77] F. Koufogiannis, S. Han, and G. J. Pappas. Gradual release of sensitive data under differential privacy. J. Priv. Confidentiality, 7(2), 2016. doi: 10.29012/jpc.v7i2.649. URL https://doi.org/10.29012/jpc.v7i2.649.

[78] N. Küchler, E. Opel, H. Lycklama, A. Viand, and A. Hithnawi. Cohere: Managing differential privacy in large scale systems. IEEE Symposium on Security and Privacy, 2024. doi: 10.48550/arXiv.2301.08517. URL https://doi.org/10.48550/arXiv.2301.08517.

[79] M. Lécuyer. Practical privacy filters and odometers with rényi differential privacy and applications to differentially private deep learning. CoRR, abs/2103.01379, 2021. URL https://arxiv.org/abs/2103.01379.

[80] M. Lécuyer, R. Spahn, K. Vodrahalli, R. Geambasu, and D. Hsu. Privacy accounting and quality control in the sage differentially private ML platform. In T. Brecht and C. Williamson, editors, Proceedings of the 27th ACM Symposium on Operating Systems Principles, SOSP 2019, Huntsville, ON, Canada, October 27-30, 2019, pages 181–195. ACM, 2019. doi: 10.1145/3341301.3359639. URL https://doi.org/10.1145/3341301.3359639.

[81] C. Li, M. Hay, G. Miklau, and Y. Wang. A data- and workload-aware query answering algorithm for range queries under differential privacy. Proc. VLDB Endow., 7(5):341–352, 2014. doi: 10.14778/2732269.2732271. URL http://www.vldb.org/pvldb/vol7/p341-li.pdf.

[82] C. Li, G. Miklau, M. Hay, A. McGregor, and V. Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. VLDB J., 24(6):757–781, 2015. doi: 10.1007/s00778-015-0398-x. URL https://doi.org/10.1007/s00778-015-0398-x.

[83] K. Ligett, S. Neel, A. Roth, B. Waggoner, and Z. S. Wu. Accuracy first: Selecting a differential privacy level for accuracy constrained erm. In Neural Information Processing Systems, 2017. URL https://api.semanticscholar.org/CorpusID:19294675.

[84] T. Luo, M. Pan, P. Tholoniat, A. Cidon, R. Geambasu, and M. Lécuyer. Privacy budget scheduling. In A. D. Brown and J. R. Lorch, editors, 15th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2021, July 14-16, 2021, pages 55–74. USENIX Association, 2021. URL https://www.usenix.org/conference/osdi21/presentation/luo.

[85] X. Luo, J. Pei, Z. Cong, and C. Xu. On shapley value in data assemblage under independent utility. Proc. VLDB Endow., 15(11):2761–2773, 2022. doi: 10.14778/3551793.3551829. URL https://www.vldb.org/pvldb/vol15/p2761-luo.pdf.

[86] X. Luo, J. Pei, C. Xu, W. Zhang, and J. Xu. Fast shapley value computation in data assemblage tasks as cooperative simple games. Proc. ACM Manag. Data, 2(1):56:1–56:28, 2024. doi: 10.1145/3639311. URL https://doi.org/10.1145/3639311.

[87] M. Mazmudar, T. Humphries, J. Liu, M. Rafuse, and X. He. Cache me if you can: Accuracy-aware inference engine for differentially private data exploration. Proc. VLDB Endow., 16(4):574–586, 2022. URL https://www.vldb.org/pvldb/vol16/p574-mazmudar.pdf.

[88] F. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In U. Çetintemel, S. B. Zdonik, D. Kossmann, and N. Tatbul, editors, Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009, pages 19–30. ACM, 2009. doi: 10.1145/1559845.1559850. URL `https://doi.org/10.1145/1559845.1559850`.

[89] P. Mohan, A. Thakurta, E. Shi, D. Song, and D. E. Culler. GUPT: privacy preserving data analysis made easy. In K. S. Candan, Y. Chen, R. T. Snodgrass, L. Gravano, and A. Fuxman, editors, Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012, pages 349–360. ACM, 2012. doi: 10.1145/2213836.2213876. URL `https://doi.org/10.1145/2213836.2213876`.

[90] P. Nanayakkara, J. Bater, X. He, J. Hullman, and J. Rogers. Visualizing privacy-utility trade-offs in differentially private data releases. Proc. Priv. Enhancing Technol., 2022(2):601–618, 2022. doi: 10.2478/POPETS-2022-0058. URL `https://doi.org/10.2478/popets-2022-0058`.

[91] M. Nasr, J. Hayes, T. Steinke, B. Balle, F. Tramèr, M. Jagielski, N. Carlini, and A. Terzis. Tight auditing of differentially private machine learning. In J. A. Calandrino and C. Troncoso, editors, 32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023, pages 1631–1648. USENIX Association, 2023. URL `https://www.usenix.org/conference/usenixsecurity23/presentation/nasr`.

[92] J. P. Near and X. He. Differential privacy for databases. Found. Trends Databases, 11(2):109–225, 2021. doi: 10.1561/1900000066. URL `https://doi.org/10.1561/1900000066`.

[93] J. P. Near, D. Darais, C. Abuah, T. Stevens, P. Gaddamadugu, L. Wang, N. Somani, M. Zhang, N. Sharma, A. Shan, et al. Duet: an expressive higher-order language and linear type system for statically enforcing differential privacy. Proceedings of the ACM on Programming Languages, 3(OOPSLA):1–30, 2019.

[94] P. Nguyen, A. Silence, D. Darais, and J. P. Near. Duetsgx: Differential privacy with secure hardware. CoRR, abs/2010.10664, 2020. URL `https://arxiv.org/abs/2010.10664`.

[95] K. Nissim, S. Raskhodnikova, and A. D. Smith. Smooth sensitivity and sampling in private data analysis. In D. S. Johnson and U. Feige, editors, Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007, pages 75–84. ACM, 2007. doi: 10.1145/1250790.1250803. URL `https://doi.org/10.1145/1250790.1250803`.

[96] OpenDP. Developing open source tools for differential privacy. `https://docs.opendp.org/en/stable/index.html`. Accessed: 2024-05-31.

[97] B. Pan, N. Stakhanova, and S. Ray. Data provenance in security and privacy. ACM Comput. Surv., 55(14s):323:1–323:35, 2023. doi: 10.1145/3593294. URL `https://doi.org/10.1145/3593294`.

[98] P. Pappachan, S. Zhang, X. He, and S. Mehrotra. Don't be a tattle-tale: Preventing leakages through data dependencies on access control protected data. Proc. VLDB Endow., 15(11):2437–2449, 2022. URL `https://www.vldb.org/pvldb/vol15/p2437-pappachan.pdf`.

[99] P. Pappachan, S. Zhang, X. He, and S. Mehrotra. Preventing inferences through data dependencies on sensitive data. IEEE Transactions on Knowledge and Data Engineering, to appear. URL `https://doi.org/10.1109/TKDE.2023.3336630`.

[100] S. Peng, Y. Yang, Z. Zhang, M. Winslett, and Y. Yu. Query optimization for differentially private data management systems. In C. S. Jensen, C. M. Jermaine, and X. Zhou, editors, 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013, pages 1093–1104. IEEE Computer Society, 2013. doi: 10.1109/ICDE.2013.6544900. URL https://doi.org/10.1109/ICDE.2013.6544900.

[101] K. Pillutla, G. Andrew, P. Kairouz, H. B. McMahan, A. Oprea, and S. Oh. Unleashing the power of randomization in auditing differentially private ML. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/d09ef5264966e17adffd3157265c9946-Abstract-Conference.html.

[102] D. Proserpio, S. Goldberg, and F. McSherry. Calibrating data to sensitivity in private data analysis. Proc. VLDB Endow., 7(8):637–648, 2014. doi: 10.14778/2732296.2732300. URL http://www.vldb.org/pvldb/vol7/p637-proserpio.pdf.

[103] D. Pujol, Y. Wu, B. Fain, and A. Machanavajjhala. Budget sharing for multi-analyst differential privacy. Proc. VLDB Endow., 14(10):1805–1817, 2021. doi: 10.14778/3467861.3467870. URL http://www.vldb.org/pvldb/vol14/p1805-pujol.pdf.

[104] D. Pujol, A. Sun, B. Fain, and A. Machanavajjhala. Multi-analyst differential privacy for online query answering. Proc. VLDB Endow., 16(4):816–828, 2022. URL https://www.vldb.org/pvldb/vol16/p816-pujol.pdf.

[105] C. R. Rao. Sufficient statistics and minimum variance estimates. In Mathematical Proceedings of the Cambridge Philosophical Society, volume 45, pages 213–218. Cambridge University Press, 1949.

[106] C. Ré and D. Suciu. Approximate lineage for probabilistic databases. Proceedings of the VLDB Endowment, 1(1):797–808, 2008.

[107] R. Rogers, S. Subramaniam, S. Peng, D. Durfee, S. Lee, S. K. Kancha, S. Sahay, and P. Ahammad. Linkedin's audience engagements api: A privacy preserving data analytics system at scale. arXiv preprint arXiv:2002.05839, 2020.

[108] R. M. Rogers, S. P. Vadhan, A. Roth, and J. R. Ullman. Privacy odometers and filters: Pay-as-you-go composition. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, pages 1921–1929, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/58c54802a9fb9526cd0923353a34a7ae-Abstract.html.

[109] A. Roy Chowdhury, C. Wang, X. He, A. Machanavajjhala, and S. Jha. Crypte: Crypto-assisted differential privacy on untrusted servers. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pages 603–619, 2020.

[110] A. D. Smith. Privacy-preserving statistical estimation with optimal convergence rates. In L. Fortnow and S. P. Vadhan, editors, Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011, pages 813–822. ACM, 2011. doi: 10.1145/1993636.1993743. URL https://doi.org/10.1145/1993636.1993743.

[111] Snowflake Inc. Differential privacy in Snowflake data clean rooms. `https://docs.snowflake.com/en/user-guide/cleanrooms/differential-privacy`. Accessed: 2024-05-31.

[112] D. Srivastava and Y. Velegrakis. Intensional associations between data and metadata. In Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pages 401–412, 2007.

[113] D. Sun, W. Dong, and K. Yi. Confidence intervals for private query processing. Proc. VLDB Endow., 17 (3):373–385, 2023. URL `https://www.vldb.org/pvldb/vol17/p373-sun.pdf`.

[114] J. Tang, A. Korolova, X. Bai, X. Wang, and X. Wang. Privacy loss in apple's implementation of differential privacy on macos 10.12. Theory and Practice of Differential Privacy (TPDP) Workshop, 2017.

[115] Y. Tao, A. Gilad, A. Machanavajjhala, and S. Roy. Dpxplain: Privately explaining aggregate query answers. Proc. VLDB Endow., 16(1):113–126, 2022. doi: 10.14778/3561261.3561271. URL `https://www.vldb.org/pvldb/vol16/p113-tao.pdf`.

[116] P. Thaker, M. Budiu, P. Gopalan, U. Wieder, and M. Zaharia. Overlook: Differentially private exploratory visualization for big data. arXiv preprint arXiv:2006.12018, 2020.

[117] Transcend Inc. Next-gen privacy management. `https://transcend.io/`. Accessed: 2024-05-31.

[118] TUMULT Labs. Tumult analysis. `https://www.tmlt.dev/`. Accessed: 2024-05-31.

[119] S. Vadhan and T. Wang. Concurrent composition of differential privacy. In Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part II 19, pages 582–604. Springer, 2021.

[120] S. Vadhan and W. Zhang. Concurrent composition theorems for differential privacy. In Proceedings of the 55th Annual ACM Symposium on Theory of Computing, pages 507–519, 2023.

[121] E. L. Vesga, A. Russo, and M. Gaboardi. A programming framework for differential privacy with accuracy concentration bounds. In 2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020, pages 411–428. IEEE, 2020. doi: 10.1109/SP40000.2020.00086. URL `https://doi.org/10.1109/SP40000.2020.00086`.

[122] S. Wagh, X. He, A. Machanavajjhala, and P. Mittal. Dp-cryptography: marrying differential privacy and cryptography in emerging applications. Communications of the ACM, 64(2):84–93, 2021.

[123] Y. Wang, Z. Ding, G. Wang, D. Kifer, and D. Zhang. Proving differential privacy with shadow execution. In Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, pages 655–669, 2019.

[124] Y. Wang, Z. Ding, D. Kifer, and D. Zhang. Checkdp: An automated and integrated approach for proving differential privacy or finding precise counterexamples. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pages 919–938, 2020.

[125] Y. Wang, Z. Ding, Y. Xiao, D. Kifer, and D. Zhang. Dpgen: Automated program synthesis for differential privacy. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pages 393–411, 2021.

[126] J. Whitehouse, A. Ramdas, Z. S. Wu, and R. M. Rogers. Brownian noise reduction: Maximizing privacy subject to accuracy constraints. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, Advances in Neural Information Processing Systems 35: Annual Conference on

Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/48aaa5ea741ae8430bd58e25917d267d-Abstract-Conference.html.

[127] J. Whitehouse, A. Ramdas, R. Rogers, and S. Wu. Fully-adaptive composition in differential privacy. In International Conference on Machine Learning, pages 36990–37007. PMLR, 2023.

[128] R. J. Wilson, C. Y. Zhang, W. Lam, D. Desfontaines, D. Simmons-Marengo, and B. Gipson. Differentially private SQL with bounded user contribution. Proc. Priv. Enhancing Technol., 2020(2):230–250, 2020. doi: 10.2478/popets-2020-0025. URL https://doi.org/10.2478/popets-2020-0025.

[129] D. Winograd-Cort, A. Haeberlen, A. Roth, and B. C. Pierce. A framework for adaptive differential privacy. Proceedings of the ACM on Programming Languages, 1(ICFP):1–29, 2017.

[130] Y. Wu, V. Tannen, and S. B. Davidson. Provenance-based model maintenance: Implications for privacy. IEEE Data Eng. Bull., 45(1):37–49, 2022. URL http://sites.computer.org/debull/A22mar/p37.pdf.

[131] X. Xiao, G. Bender, M. Hay, and J. Gehrke. ireduct: differential privacy with reduced relative errors. In T. K. Sellis, R. J. Miller, A. Kementsietsidis, and Y. Velegrakis, editors, Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011, pages 229–240. ACM, 2011. doi: 10.1145/1989323.1989348. URL https://doi.org/10.1145/1989323.1989348.

[132] Y. Xiao, Z. Ding, Y. Wang, D. Zhang, and D. Kifer. Optimizing fitness-for-use of differentially private linear queries. Proc. VLDB Endow., 14(10):1730–1742, 2021. doi: 10.14778/3467861.3467864. URL http://www.vldb.org/pvldb/vol14/p1730-xiao.pdf.

[133] Y. Xiao, G. Wang, D. Zhang, and D. Kifer. Answering private linear queries adaptively using the common mechanism. CoRR, abs/2212.00135, 2022. doi: 10.48550/arXiv.2212.00135. URL https://doi.org/10.48550/arXiv.2212.00135.

[134] L. Yu, S. Zhang, L. Zhou, Y. Meng, S. Du, and H. Zhu. Thwarting longitudinal location exposure attacks in advertising ecosystem via edge computing. In 42nd IEEE International Conference on Distributed Computing Systems, ICDCS 2022, Bologna, Italy, July 10-13, 2022, pages 470–480. IEEE, 2022. doi: 10.1109/ICDCS54860.2022.00052. URL https://doi.org/10.1109/ICDCS54860.2022.00052.

[135] D. Zhang and D. Kifer. Lightdp: Towards automating differential privacy proofs. In Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, pages 888–901, 2017.

[136] D. Zhang, R. McKenna, I. Kotsogiannis, M. Hay, A. Machanavajjhala, and G. Miklau. EKTELO: A framework for defining differentially-private computations. In G. Das, C. M. Jermaine, and P. A. Bernstein, editors, Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018, pages 115–130. ACM, 2018. doi: 10.1145/3183713.3196921. URL https://doi.org/10.1145/3183713.3196921.

[137] S. Zhang. DProvSQL: Accuracy-aware privacy provenance framework for differentially private sql engine. Master's thesis, University of Waterloo, 2022.

[138] S. Zhang and X. He. DProvDB: Differentially private query processing with multi-analyst provenance. In Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2024. ACM, 2024.

[139] S. Zhang, X. He, A. Kundu, S. Mehrotra, and S. Sharma. Secure normal form: Mediation among cross cryptographic leakages in encrypted databases. In IEEE International Conference on Data Engineering (ICDE). IEEE, 2024.

[140] W. Zhang, O. Ohrimenko, and R. Cummings. Attribute privacy: Framework and mechanisms. In FAccT '22: 2022 ACM Conference on Fairness, Accountability, and Transparency, Seoul, Republic of Korea, June 21 - 24, 2022, pages 757–766. ACM, 2022. doi: 10.1145/3531146.3533139. URL https://doi.org/10.1145/3531146.3533139.

Transparent Decisions: Selective Information Disclosure To Generate Synthetic Data

Carlos Gavidia-Calderon[1], Steve Harris[2],[3], Markus Hauru[1], Florimond Houssiau[1], Carsten Maple[1,4], Iain Stenson[1] and May Yong[1]

[1]The Alan Turing Institute
[2]University College London
[3]NIHR University College London Hospitals BRC
[4]University of Warwick

Abstract

The UK government and the public wish to see the National Health Service (NHS) use data and Artificial Intelligence for public good [13][16]. However, there is a major challenge in making health data available for research whilst respecting patient privacy. Synthetic data generation is an emerging technique that enables access to data that, in some way, shares the characteristics of the original data. In this paper we introduce SqlSynthGen (SSG), a method for generating synthetic relational datasets. SSG offers a human-readable, risk-guided approach to refining data fidelity while managing disclosure risk. This paper presents SSG, specifically focusing on its application for generating synthetic data from NHS hospitals.

1 Introduction

Hospitals electronic health record systems are typically built using relational databases containing millions of records. While hospital staff access this data for their clinical duties, other professional communities— scientists, software engineers and educators — rightly must follow lengthy processes to be granted access. Controls are in place to ensure patient data —which is both sensitive and valuable [28]— is accessed for only legitimate reasons. Current practices involve preparing employee contracts, implementing de-identification or anonymisation mechanisms to remove personal information, and accessing data only via Trusted Research Environments [14].

While protecting patient privacy is of utmost importance, these processes impede collaboration and engagement, and introduce delays to researchers already working to arduous grant deadlines. For instance, researchers can use data to improve diagnostic accuracy, refine our understanding of diseases, or develop personalised treatments [30]. Patient data can be used to train the next generation of healthcare practitioners and researchers. Synthetic data is an accelerator: it can provide a simulcrum with the characteristics of patient data that can be shared onwardly. This can be used to support education and training, to quality control applications and code, and to test reproducible analytical pipelines in the open. This will accelerate academic progress for patient benefit.

In order to both protect user privacy and control access, current techniques employ mechanisms including data agreements, de-identification or anonymisation, aggregation over the original data, and provision of trusted

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

research environments (TRE) for access by third parties. While these techniques provide an extra layer of protection, they are not immune to vulnerabilities [21]. For example, de-identified data releases are still susceptible to linkage attacks. Aggregation requires releasing only aggregate population metrics, such as counts or averages, but outliers remain vulnerable to identification [30]. Instead of releasing real patient data —either partial or aggregate— an option is to release synthetic patient data.

Synthetic data is data that is manufactured, as opposed to real data that is collected from real-life events and people. Synthetic data generators (SDG) use algorithms to produce synthetic data entries while preserving statistical properties of the real dataset. There are multiple SDG approaches in the literature, each one targeting a specific data type, such as tabular data or time-series data [17]. SDGs can, when appropriately constructed, offer mathematical guarantees of the preservation of user privacy [19, 8] by incorporating differential privacy.

In this paper, we describe our work on developing a new SDG approach at the University College London Hospitals (UCLH) NHS Foundation Trust. Each year, UCLH admits 100,000 patients and stores their data in a relational database. Broadly, we discover that these are their requirements regarding their utilisation:

- **REQ-1:** The synthetic datasets should be in the form of relational datasets for any given relational schema
- **REQ-2:** The generator can manufacture synthetic data by utilising aggregates and statistical properties extracted from real patients
- **REQ-3:** Ensure that information disclosed about real patients are easily understandable by humans.

Listing 1: Requirements for Synthetic Data Generation at UCLH Trust

We developed SqlSynthGen [12] to meet the requirements in Listing 1. SqlSynthGen is an open-source Python package that can replicate the database schema of a relational database. Once the replica is in place, SqlSynthGen can generate synthetic samples at different levels of fidelity: from low-fidelity random values compliant with the database schema, to high-fidelity samples from probability distributions learned from real data.

SqlSynthGen uses a white-box approach where information extraction from real data are expressed as SQL queries in human-readable format, rather than black-box approaches, such as deep generative models with thousands of parameters [6]. For ensuring patient privacy, SqlSynthGen supports differential privacy (DP)[10] to add quantifiable noise to the information extracted from the real data.

2 Sharing Patient Data

This section starts by enumerating motivations for sharing patient data. An understanding of motivations is important because these determine the requirements of appropriate data sharing mechanisms. The reasoning for sharing data dictates what minimum data needs to be shared, and this in turn defines the requirements to be met if the data is to be shared reasonably safely.

We then survey the current privacy preservation practices currently adopted by hospitals to enable collaborators controlled access to hospital data. We show that these are a) linked to inadequate privacy protection measures [21, 30], or b) a cause of unnecessary friction to analysis [23]. While synthetic data is considered a potential solution to overcome the above challenges, many patient datasets are organised as relational databases. Current synthetic data generators have limitations: a) they do not address the unique challenges of the relational structures [22][32]; b) they require users to specify dataset schemas [29]; or c) they can achieve differentially private, explainable, high-fidelity synthetic data for relational databases but currently face limitations in scalability. [8].

2.1 On the Benefits of Sharing Patient Data

Enhancing Research Quality and Innovation: Collaboration can lead to more comprehensive research studies, allowing healthcare practitioners and researchers to test hypotheses or observe trends across a broader dataset than is available internally. How well a dataset represents the true distribution matters more than simply dataset size[2]. In the medical domain, where lack of data is a common occurrence, the amalgamation of diverse datasets has a better chance of representing true underlying distributions.

Access to Specialised Expertise: External collaborators bring specialised knowledge and skills that complement the in-house capabilities of a hospital. For example, collaborations with methodology researchers can lead to state-of-the-art data analysis and interpretation, thereby improving both method development and treatment outcomes. Software engineers and machine learning operations engineers can build customised cyber-physical infrastructure to support analysis of patient data in real time[14].

Accelerating Medical Discoveries: By pooling resources and data between hospitals, research can proceed at a faster pace[2], potentially leading to quicker discoveries in disease mechanisms, treatment effectiveness, and development of new therapies or medical technologies. Sharing patient data can facilitate the recruitment of participants for clinical trials, ensuring a diverse and adequate sample size. This can be crucial in studying rare diseases or sub-types of common diseases, especially in hospitals that offer specialisations not commonly offered elsewhere in the world.

Expanding Research Funding Opportunities: Collaborative research often has better chances of securing funding[31]. Funding bodies frequently encourage or require collaboration across institutions as a criterion for grants, viewing it as a way to maximise the impact of their investment.

Bench-marking and Quality Improvement: Comparing data across institutions can help identify best practices and areas for improvement in patient care and management. This bench-marking is used to drive quality improvement initiatives within a hospital[33].

Education and Training: Collaborations provide educational opportunities to clinical research employees at hospitals, researchers and students at universities and research institutions, exposing them to different perspectives, methodologies, and cutting-edge research through joint ventures and knowledge exchanges.

Building Networks and Reputation: Collaborations can enhance a hospital's reputation in the medical and scientific community[31]. They extend the hospital's influence and recognition, which can attract top talent and more collaborations in the future.

2.2 Current Practices For Sharing Patient Data

De-identification and Anonymisation of Patient Data: De-identification is the process of obscuring or replacing personal identifiers to prevent the direct association of data with an individual. Common de-identification methods include explicit removal, masking or pseudonymisation of direct identifiers, and aggregating data to remove specificity eg. binning.

Anonymisation aims to ensure that data cannot be linked back to an individual by any means. Anonymisation strips datasets of all personal identifying information but it is not provable when this has been achieved. Conservative measures will strip a lot of information thereby heavily affecting the value of the dataset, and we still cannot be certain that there is not some way to de-anonymise.

For example, the removal of timestamps from a medical dataset as part of an de-identification or anonymisation process is performed because timestamps can be used to re-identify a patient by linking a patient's records over multiple de-identified datasets. The pattern of timestamps can disclose information about a patient's health, as well as their frequencies away from home.

However the stripping of timestamps from a medical dataset erases important information because medical information is highly time-contextual. Part of the richness of medical data is its time-series nature. Medical data that has been stripped of time stamps has reduced richness of data and is limited what can be learnt from it.

Effectiveness of both de-identification and anonymisation techniques is highly dependent on context, which includes the dimensionality, volume, and statistical properties of data. Other important aspects that need to be considered include which types of applications or analyses the data are to be used for, whether the data will be released publicly or with additional access control, and whether the data are tabular, relational, or have longitudinal or transactional characteristics.

Trusted Research Environments: Trusted Research Environments (TREs) are an important part of the data sharing mechanism ecosystem. TREs are the secure infrastructure and governance model that allows researchers to access and analyse data; they are often used in conjunction with other data-sharing mechanisms.

TREs play a major role in controlling data access levels. Initially, data access is controlled through secure authentication and authorisation mechanisms. This means that only approved researchers can access the data, and they can only access specific datasets approved for their role and research projects. Activities in TREs are closely monitored and logged.

In addition, TREs provide both physical and virtual security. Data in TREs are often stored in physically protected facilities. Virtual security measures such as firewalls, intrusion detection systems and regular penetration testing maximise protection against external threats. Finally, to ensure no privacy leakage, data egress from TREs is restricted. Researchers can analyse data within TREs but cannot take it out.

This means that working with data within TREs is far from a comfortable experience [23]. In order to provide security measures, computational resources can be limited and the list of approved software packages for analysis is restricted and not easily updated. There is significant process overhead generated by the need for detailed authentication into remote machines, activity logging, monitoring and compliance checks. There is a steep learning curve in working within a TRE, and new users are heavily dependent on support staff for technical assistance. Finally, the inability to egress data limits the sharing of interim findings and prevents close collaboration on ongoing data analysis.

Honorary contracts and data agreements: In order for non-hospital/clinical staff to work with medical data, they typically either need to become honorary employees of a trust or their current institution need to enter into a data sharing agreement with the trust. Both are lengthy and restrictive.

The process of obtaining an honorary contract typically begins with an initial inquiry and application to the relevant department or clinical group at the hospital. This is followed by credential verification and background checks, including border security investigations. Once these checks are satisfactorily completed, the relevant departments can grant approval.

To get a data agreement signed between two institutions, the first step is to identify the need for data sharing, specifying what data will be shared and how it will be used. Next, security requirements for storing, protecting, and accessing the data must be agreed upon by both parties. All these elements need to comply with relevant regulations. Finally, the agreement must be reviewed by the legal and compliance teams of both institutions to ensure all requirements are met and all parties are protected.

3 From Sharing Real Data to Sharing Synthetic Data

Real data is recorded from real life. Synthetic data is manufactured data, and can be created such that data elements are random, structurally or type accurate, or have distributions that mirror statistical properties of another dataset. In the last case, statistical properties can be directly or indirectly observed, to inform the data manufacturing process. When any properties of one dataset is used to guide the manufacturing process of another dataset, the first dataset is referred to as the ’real’ or ’original’ data. In the use case presented in this paper, ’real’ data is hospital patient data. Our manufactured data is commonly referred to as ’synthetic’ data.

While manufactured patient data is not about real individuals, it is a fallacy to imagine that adoption of synthetic data in data sharing practices prevents disclosure of sensitive information. This section shows how synthetic data generators can manufacture outputs which disclose more, or less sensitive information, and how this affects the ways in which outputs can be used.

3.1 Synthetic Data Generators

Synthetic data generators (SDG) manufacture data. There is a tension observed in the process of manufacturing synthetic data which involves three factors: fidelity, utility and privacy. Fidelity measures the extent to which synthetic data resembles the real dataset. Utility is the measure of the usefulness of synthetic data to a given task. Privacy is a measure of the information disclosed about the real dataset during generation of the synthetic dataset. These three factors inform the manufacturing process and limit the ways its outputs can be used. Synthetic data which is very similar to the real dataset (high fidelity) risk leaking information about real patients (low privacy). Conversely, low fidelity datasets typically contain little information relating to the real data, so individuals are unlikely to be identified. However, this low fidelity also limits the dataset’s utility. For instance, medical data stripped of personal identifiers such as timestamps loses its richness and reduces the scope of insights that can be derived from it.

However, low-fidelity or coarse-grained datasets can still be useful, as utility is dependent on the context or task. In some cases, low-fidelity datasets are valuable if they provide sufficient information for engineering applications e.g. software testing. When paired with real data, multi-fidelity datasets can reduce computational costs and prevent over-fitting in machine learning tasks [26][27][5]. Low fidelity datasets can remove blockers at the beginning of research for initial exploration, building pipelines, and testing models. These tasks can be conducted in a secure environment restricted to students and researchers, with scripts later ported to the hospital for training on real data if the initial analysis proves promising.

This means that there is a class of low-fidelity datasets that is useful in common research and engineering tasks. The benefits of using these datasets can be realised with little cost to patient privacy.

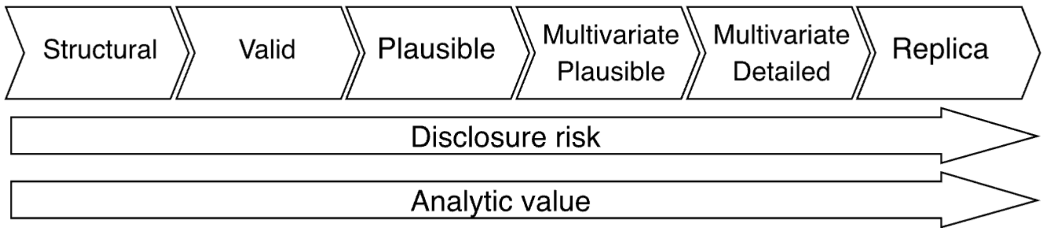


Figure 1: Shows the range of fidelity for synthetic data. High fidelity data can result in higher utility, but also increased risk of identification. Sourced from UK Office of National Statistics[24].

The UK Office of National Statistics [24] have defined a spectrum of fidelity for synthetic data, shown in Figure 1. In the context of healthcare relational datasets:

- **Structurally correct datasets** have the same column names, tables and relationships as real data.

- **Valid datasets** imply that the values in the synthetic dataset are correct and valid, e.g. date of births are valid dates.
- **Plausible datasets** imply that the relationship between values are realistic, e.g. a patient's date of death is not before their date of birth.
- **Multivariate plausible datasets** implies that the values are correlated across different variables, e.g. a male patient is likely to be both heavier and taller than a female patient.
- **Multivariate detailed datasets** are more realistic than a multivariate plausible data set, but less than a replica of the real data. An example are rows of data showing that a patient with a diabetes diagnosis has more records of blood sugar readings than a patient with a broken bone.

3.2 Synthetic Data For UCLH NHS Trust

University College London Hospitals National Health Services Foundation (UCLH NHS) Trust is a pioneering institution within the UK, renowned for its treatment care and specialist services not widely available in other NHS Trusts. It is closely affiliated with University College London; this is a partnership that emphasises research and education, integrating medical research and teaching at the undergraduate and postgraduate levels directly into the clinical environment. As an institute that emphasises medical care, research and education, and as custodians of highly sensitive medical data, UCLH NHS Trust are in a position to leverage research capabilities to supercharge innovation if they can develop a process for thoughtful access to this data. However, consequences of unintentionally releasing identifiable information include loss of individuals' privacy, loss of institutional prestige, as well as substantial legal fines.

3.2.1 Problem Statement

Machine learning (ML) infrastructure are deployed in hospitals to enable AI in healthcare delivery and administration. ML infrastructure supports tasks such as structuring data from electronic health records into a format that can be used as inputs to AI algorithms, deploying image analysis and predictive analysis tools, and presenting the results to healthcare practitioners in a timely and useful format.

To achieve these tasks, engineers who build the infrastructure need to gain an understanding of the data structures and data flow within the hospital. Researchers need to evaluate if target datasets meet their purposes for hypothesis testing, and are adequate in terms of quality and quantity. It is onerous to issue contracts to entire teams of engineers, researchers and students, but there are no other ways to share data with external collaborators.

However, what engineers and researchers need when working on early stages of exploratory analysis to understand data in terms of content, structure and data flow is information <u>about</u> the data, rather than having access to individual rows of data itself. Here is an opportunity to frame the problem as: <u>What information can be released about sensitive data, which is maximally beneficial to engineers and researchers, with minimal cost to patient privacy</u>?

3.2.2 Requirements

Listing 1 enumerates the requirements of building synthetic data generators for UCLH Trust. This section expands on each requirement; the following section demonstrates how the design of SSG fulfils these requirements.

Produce relational datasets for any given schema: Many data holders, including hospitals, store patient electronic health records in relational databases. Data is often structured within complex schema that capture both single observations and time series data. These relational databases also include tables for vocabularies such as definitions of drugs, observations and diagnoses.

Under this requirement, a minimally useful synthetic dataset must at the very least a) be structurally correct. That is, it will contain the same tables, columns, and data types as the real data, and b) meet foreign key constraints. In order to increase analytical value as shown in Figure 1, the synthetic generator will need to generate values which are valid and plausible, e.g. valid gender values and a plausible distribution of height and weight. A multivariate plausible dataset will have values that correlate across multiple tables, e.g. the correlation between gender and height are represented across the ‘Demographic‘ and ‘Observation‘ tables.

An additional complexity here is in generating synthetic time series data, e.g. blood pressure values every ten minutes for a patient in intensive care unit. In order to be multivariate plausible, the data needs to contain the correct frequencies for data collection as well as plausible values that depend on a patient’s physiology. This is generated across multiple tables as well.

Generate synthetic data using statistical properties computed from real patients Hospitals are mandated or encouraged by various information acts to release hospital information to the public. The main reasons for this are a) allowing insights into quality of care provided by public or insurance funds and b) to enable patients to make informed decisions regarding where to seek care based on hospital performance and specialisations[33].

The type of information that is released in the public domain includes quality of care indicators, patient safety data, readmission rates and service availability. This includes aggregate data about patient outcomes, infection rates, details on specialised services, bed occupancy, Accidents and Emergency (A&E) wait times as well as statistical properties on patients returning for treatment within a period of discharge. This information is published regularly and does not compromise individual patient privacy.

Synthetic data generators can use aggregate data and statistical properties of real data to generate datasets which are measurably closer to real data. A synthetic dataset generated using public information is unlikely to reveal any additional patient information beyond what is already publicly available.

Ensure that information disclosed about real patients are easily understandable by humans. Aggregates and statistical properties are well-understood mathematical concepts. A comprehensive explanation of such information extracted from real patients datasets for the purpose of generating synthetic data should cover the following three points:

1. **Extracted Information**: Detail what specific information about patients has been extracted.
2. **Computation Process**: Explain how this information is computed.
3. **Usage for Synthetic Data**: Describe how this information is used to shape the synthetic data.

Providing this explanation in a single, human-readable source ensures consistency and prevents obsolescence across multiple data generation iterations. This offers a clear audit trail of the generation process and helps identify the disclosure risks of its outputs.

The concept of synthetic data is complex, people may not understand how data that does not represent real individuals still needs privacy considerations. It is furthermore difficult to understand how the application of differential privacy to aggregates and statistics can provide additional protection.

Differential privacy (DP) [10] is the gold standard that protects individuals within a dataset while still allowing for the useful analysis of the aggregate data. Its internal mechanics of noise addition for the purpose of privacy preservation can leave users without a clear understanding of its outputs and how to interpret them correctly[9].

The application of differential privacy to synthetic data compounds the explanations’ complexities. There is a struggle to understand how DP offers probabilistic but not absolute guarantees. Explaining this to custodians of highly sensitive data is difficult because privacy is expected but not always technically feasible.

However, this is an important discussion, there is a necessary understanding to be achieved here because the interplay between privacy and utility governs the results of a differentially private synthetic data generator. The only people who can take the responsibility for managing the balance between privacy and utility are the data custodians.

4 Generating Synthetic Data Using SQLSYNTHGEN

SQLSYNTHGEN (SSG) is a software package developed to meet the requirements outlined in Section 3.3. When connected to an existing relational database, SSG builds a new empty database with the same schema. It copies over the non-sensitive data, such as look-up tables, and generates structurally correct synthetic data with random values. Optionally, SSG can refine these synthetic values using aggregates and statistical properties. SSG can apply differential privacy to obfuscate the true values of these properties in a measurable way. The new database is then populated with these synthetic values.

4.1 Technical Overview

The default output dataset from SSG is structurally correct and has no disclosure risk. These are datasets that sit on the far left end of the spectrum in Figure 1. No information about the real dataset has been disclosed, beyond the structure in which they are stored. This can already be useful e.g. for building software testing modules and pipe-lining scripts, and can be safely released if vocabularies and schema can be shared. ***This meets REQ-1: Produce relational datasets for any given schema.***

SSG can be further configured to generate synthetic data that (in reference to Figure 1), can be as sophisticated as multivariate plausible data. This is achieved by allowing the user to define SQL statements that extract aggregate statistics and statistical properties from the real data. These extracted values are then used to shape the distributions and marginals of the synthetic data. ***This meets REQ-2: To generate synthetic data using statistical properties computed from real patients.***

As part of its process, SSG generates a human-readable audit trail that details the entire data generation process. This includes what information was extracted from real data, the methods used for extraction, the computed results, and how these values were injected into the synthetic data generation. The audit trail is a human readable file whose contents are incorporated directly into the SDG process. ***Ensure that information disclosed about real patients are easily understandable by humans.***

SSG pipeline design enables the selective production of synthetic datasets with varying levels of fidelity. Users control the shaping of synthetic data by specifying which information is extracted from real data, how it is computed, and how it is utilised. SSG's configuration supports agile development, allowing for incremental fidelity improvements as needed, while maintaining transparency, auditability, and control over privacy risks at every stage. Additionally, users have the option to apply differential privacy to protect the marginals extracted from the source data.

In order to support this design, SSG's process for generating synthetic relational datasets can be broken into three separate steps, as shown in Figure 2. They are as follows:

1. SSG **builds** a new database to store synthetic data. This new database will be populated by synthetic data generated in the next steps. Look-up tables which do not have any privacy concerns are copied over entirely, to maintain foreign key constraints.

2. By default, SSG **generates** random but structurally correct data.

3. As an option, SSG can **refine** random values for higher accuracy by using extracted statistics from real data, with or without DP. For example, mean of height by age and gender can be extracted from real patients and the correlation be used to generate higher fidelity data.

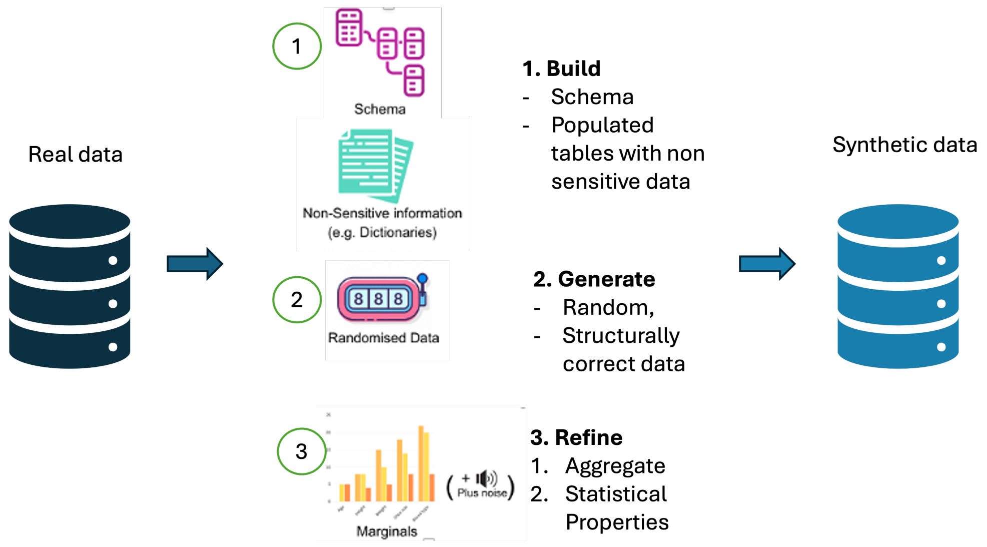


Figure 2: The processes of SQLSynthGen in order

For more information and tutorials about SQLSynthGen, please refer to our repository at `https://github.com/alan-turing-institute/sqlsynthgen`. Our repository [12] contains installation instructions, comprehensive documentation and trouble shooting guides to help get started with the software. The repository also contains a simple tutorial using a Kaggle dataset [7] as well as an advanced example based on the Observational Medical Outcomes Partnership (OMOP)[25], which provides a standardised data model for observational healthcare data.

In the following sections, we demonstrate the use of SSG in creating synthetic data based on a publicly available AirBnB Kaggle dataset [7].

4.2 Building a Replica of a Real Dataset

In this example, let us consider that our dataset is contained in a database called ‘airbnb‘ in a local PostgreSQL instance. We want to port the schema to a new ‘airbnb_synthetic‘ database, and populate the ‘airbnb_synthetic‘ database with synthetic rows that mirror some of the statistical properties of the ‘airbnb‘ dataset.

Build schema tables: We connect to the real dataset by setting connection credentials in environment variables. We run a series of commands `sqlsynthgen make-tables`, `sqlsynthgen create-tables` and `sqlsynthgen make-generators` to auto-generate two Python files.

The first file, ‘orm.py‘, outlines the structure of the PostgreSQL ‘airbnb‘ dataset by mapping each table in ’airbnb’ to a corresponding Python class. Each column in these tables is represented as a class field. This mapping is generated using SQLAlchemy[4], which is a SQL toolkit and Object-Relational Mapping (ORM) library for Python. By using SQLAlchemy in SSG for mapping, users do not need to perform any additional configuration to describe the schema of the real dataset. The ‘orm.py‘ file serves as a foundation for building a new ‘airbnb_synthetic‘ PostgreSQL database, complete with the necessary tables, columns and data types. Listing 2 shows a snippet from ‘orm.py‘ that demonstrates how the ‘users‘ table from the ‘airbnb‘ dataset is mapped as a Python class.

```python
class User(Base):
    __tablename__ = "users"

    id = Column(String, primary_key=True)
    date_account_created = Column(Date)
    ...
```

Listing 2: Section of PostgreSQL table ‘user‘ represented as a Python class

Copy over lookup tables: A lookup table, or a vocabulary, is a table used to store a predefined set of values that are referenced by other tables. They contain a finite and static set of values such as codes, names of categories or descriptions. Look-up tables are a good practice adopted help normalise databases by removing redundancy and enabling efficient data management. They work by using foreign key constraints to ensure values in related tables are consistent and valid. These foreign key constraints need to be satisfied when generating synthetic data in relational datasets. On their own, vocabularies provide only limited utility, since the more interesting aspects of the data are usually found in the non-vocabulary tables.

The fidelity of the synthetic dataset can be improved by ensuring the vocabulary tables have perfect fidelity from the beginning, since they do not raise privacy concerns (although some vocabularies are copyright-protected). In this section, we demonstrate how SSG addresses vocabulary tables by copying them in their entirety, thereby eliminating the need for synthesis.

First we specify vocabulary tables in a `config.yaml`; the listing 3 below denotes ‘countries‘ as a vocabulary table. All values in denoted vocabulary tables are copied to an auto-generated `.yaml` file. Listing 4 shows a snippet of data from the ‘countries‘ table which has been copied to a auto-generated `countries.yaml` file.

```yaml
tables:
    countries:
        vocabulary_table: true
```

Listing 3: A yaml section to demarcate table ’countries’ as a vocabulary table

```yaml
- country_destination: AU
  destination_km2: 7741220
  destination_language: eng
    :
- country_destination: CA
  destination_km2: 9984670
  destination_language: eng
  distance_km: 2828.1333
    :
```

Listing 4: Example of data rows copied from ‘countries‘ vocabulary table

The primary reason for copying vocabularies this way is to maximise transparency for auditing purposes. Data holders can audit each value extracted from the real dataset, before creating any synthetic data. Note that we have to be careful in making sure that the tables marked as vocabulary tables truly do not hold privacy sensitive data, otherwise catastrophic privacy leaks are possible, where the original data is exposed raw and in full.

The downside of this approach is clear when scaling up to address vocabulary tables which are very large. Therefore our generator pipeline is modular to ensure that vocabularies need only be copied once when creating more rows to add into a synthetic dataset.

Generate Random Values that are Structurally Correct: The second auto-generated file, ‘ssg.py‘, contains Python code that generates random values matching the data types defined by the Python classes. This human-readable Python code serves as part of the audit trail, demonstrating how values for populating each table column are generated. For complex schemas with multiple tables and columns, the generator code for each column is easily identifiable and can be customised independently of rest of the generator.

Listing 5 demonstrates the auto-generated Python code for generating ‘id‘ and ‘date_account_created‘ values for the ‘User‘ table. ‘id‘ is assigned generic, password-like values, and ‘date_account_created‘ is assigned a random date value.

```
class usersGenerator:
    num_rows_per_pass = 1

    def __init__(self, src_db_conn, dst_db_conn):
        pass
        self.id = generic.person.password()
        self.date_account_created = generic.datetime.date()
        ...
```

Listing 5: A Python class for generating synthetic id and date_account_created values for Postgres table ‘User‘

Refine values using aggregate statistics: The default behaviour of SSG is to generate syntactically correct, random values. This section shows how we incorporate aggregate and statistical properties of real data in order to generate synthetic data that retain those properties.

We demonstrate an example to generate normally distributed synthetic values to populate a ‘users.age‘ column, with reference to the mean and standard deviation values of the real data. The user begins by defining SQL statements in the ‘age_stats‘ section of a ‘config.yaml‘ file. This is demonstrated in listing 6. SSG uses the credentials provided to authenticate to the database and execute SQL statements to compute the required values. Computed values are recorded in an auto-generated `src-stats.yaml` file, demonstrated in listing 7. These can be can be referenced by the Python data generators. Listing 8 shows the Python provider function that generates a distribution of values to meet the statistical properties computed and recorded in ‘config.yaml‘ and ‘src-stats.yaml‘.

```yaml
src-stats:
    - name: age_stats
    query: >
    SELECT AVG(age)::float AS mean, STDDEV(age)::float AS std_dev
    FROM users
    WHERE age <= 100
tables:
    users:
        row_generators:
          - name: airbnb_generators.user_age_provider
            kwargs:
              query_results: SRC_STATS["age_stats"]
            columns_assigned: age
```

Listing 6: A section of the config.yaml file that shows an SQL statement to compute mean and average of column ‘users.age‘. Results are stored as ‘age_stats‘.

```yaml
age_stats:
- mean: 36.54434029695572
  std_dev: 11.708339792587486
```

Listing 7: Example of mean and standard deviation values computed from ‘users.age‘ column

```python
  import random
  def user_age_provider(query_results):
      mean: float = query_results[0]["mean"]
      std_dev: float = query_results[0]["std_dev"]
      return random.gauss(mean, std_dev)
```

Listing 8: A provider function

The primary reason for extracting information using SQL statements and documenting it in ‘config.yaml‘ is to maximise transparency for auditing purposes. Similar to vocabularies, users can audit information that is disclosed about real data by reviewing the human-readable ‘config.yaml‘ and ‘src-stats.yaml‘ files. Multiple properties, such as marginals, percentiles, and skewness, can be used simultaneously to enhance the fidelity of synthetic data. These computations can be resource-intensive with large datasets. To address this, the SSG generator process is modularised: properties are computed and stored once, allowing subsequent generators to reference these values, which will be reliable provided the real dataset has not changed significantly.

Introduce differential privacy into aggregate statistics: Differential privacy is arguably the most popular technique for providing privacy guarantees on SDGs. Let us imagine two datasets:

- A synthetic dataset B generated with information of person X.

- A synthetic dataset A generated without information of person X.

If both datasets were generated using a differentially-private mechanism, performing a query on dataset A should provide the same, or almost the same, result as performing the same query on dataset B [19]. Differentially private mechanisms hide the presence or absence of person X —or one any individual— in the dataset, which implies strong protection of their privacy [21]. To accomplish this, these mechanisms inject random noise to the synthetic data. The amount of noise is a function of the privacy parameter epsilon ε that measures how similar the datasets A and B are required to be. ε needs to be chosen carefully to provide the required privacy guarantee.

One of the most common fundamental techniques for generating synthetic data in a differentially private involved 3 steps: 1) select, or choose, some queries over the original data, 2) measure, or execute, those queries using a differentially private mechanism, and 3) generate synthetic data using these measurements [20].

SQLSYNTHGEN enables the select and measure steps by supporting differentially private SQL queries in 'src-stats.yaml' (Listing 9).

```
src-stats:
  - name: age_stats
    dp-query: >
      SELECT AVG(age) AS mean, STDDEV(age) AS std_dev
      FROM query_result
    epsilon: 0.5
    delta: 0.000001
    snsql-metadata:
      max_ids: 1
      id:
        type: string
        private_id: true
      age:
        type: float
        lower: 0
        upper: 100
```

Listing 9: A differentially-private SQL query.

Internally, SQLSYNTHGEN uses SMARTNOISE SQL [1] to execute differentially private queries. As seen in Listing 9, SMARTNOISE SQL needs additional information besides the SQL query for applying a differentially private mechanism, including the privacy parameter epsilon ε. Regarding the final generate step, the query results are made available to provider functions —demonstrated in Listing 8— so SQLSYNTHGEN users can use these measures for data generation.

5 Discussion

The proliferation of research on synthetic data over the past five years underscores its significance in addressing data scarcity and sensitivity issues in machine learning. With 25,600 papers published from 2023 to mid-2024 alone, these studies span diverse domains, including computer vision, natural language processing, and healthcare [11], primarily focusing on the generation, evaluation, and application of synthetic data, particularly using GANs [3]. Originally research-driven, these methods are now being translated into practical applications, revealing new challenges and considerations [18].

Our development of a Synthetic Data Generator (SDG) for sharing sensitive hospital information has highlighted these key challenges:

There is a lack of generators developed for relational data: The development of synthetic generators commonly explore image, text data, or tabular data. Our experience is that synthetic data generators overlook the relational data format, possibly because of the foreign key constraints satisfaction criteria. This is a problem because hospital datasets are often stored in relational formats.

There is a lack of explainability in privacy preserving mechanisms: Explainability in synthetic data generators is a crucial issue for custodians of sensitive data, especially in hospitals. The lack of explainability undermines discussions between hospital data stakeholders, including both staff and patients. One discussion impacted by the lack of explainability is that of maintaining a balance between privacy guarantees and the utility of the synthetic data. While ensuring that synthetic data generators do not leak sensitive information is essential, explaining the privacy preservation mechanisms involved can be complex. Furthermore, the processes used by generators based on GANs and deep neural networks are opaque, making it difficult to assure stakeholders of the synthetic data's reliability and safety. Finally, both generators and metrics (e.g., fidelity, diversity) used to evaluate the quality of synthetic data are not easily interpretable.

We specifically addressed this explainability challenge in a series of workshops with patient and public involvement, and using SSG as an exemplar. There were two key messages from our stakeholders. Firstly, they were reassured to understand the distinction in the source of the data. Anonymised data is processed from the original data whereas synthetic data is generated *de novo*. Secondly, they valued using a language that talked about sharing information (with synthetic data) in contrast to sharing data (with anonymisation). There was recognition that information is *already* shared and tools like SSG are trustworthy because they are transparent about what information is used to generate the synthetic data.

Despite its design to address these challenges, our SqlSynthGen tool has several limitations:

Lack of Autonomous Model Discovery: Unlike GANs-based [3] or Bayesian-based [8] generators, SSG cannot autonomously discover underlying models or relationships. Users must predetermine the models, limiting the tool's adaptability and the transferability of algorithms trained on its outputs to real-world data.

Need to Ensure Security: The design of SSG includes copying vocabulary tables in their entirety and executing SQL statements on real data based on user configurations, makes it a powerful tool. However, these features introduce risks of user errors. Accidental copying tables with sensitive data could lead to severe data breeches. Executing SQL statements without proper access controls could damage real patient information.

Lack of Evaluation: SSG allows users to selectively disclose information used to shape synthetic data outputs but it lacks an integrated evaluation mechanism. Since each piece of information is independently disclosed, there is an opportunity here to iteratively fine-tune the balance between fidelity and privacy by combining SSG with an evaluation tool such as TAPAS [15].

6 Conclusion

The number of research papers on synthetic data has surged significantly, indicating its growing importance in addressing data scarcity and sensitivity issues in machine learning. There is a notable gap in the development of synthetic data generators specifically for relational data structures. Most exciting developments on generators focus on time-series, graph, audio, imaging or tabular data structures, often neglecting the complexities associated with relational databases, such as foreign key constraints. This limitation is significant because many practical applications, particularly in healthcare, rely heavily on relational data formats.

Aside from the oversight in provision for relational data, the lack of explainability in privacy-preserving mechanisms is a critical challenge. For synthetic data to be trusted and widely adopted, especially in sensitive domains like healthcare, stakeholders need to understand how privacy is preserved. The opacity of deep learning models and GANs currently used in generating synthetic data makes it difficult to provide this assurance, which can hinder stakeholder discussions and acceptance.

The direction for future work on the application of synthetic data generation in sensitive data context is clear:

1. **Development of Relational Data Generators:** There is a clear need for synthetic data generators that can handle relational data formats effectively, addressing issues like foreign key constraints.

2. **Improving Explainability:** Enhancing the explainability of synthetic data generation processes will be crucial for gaining stakeholder trust and facilitating broader adoption by custodians of sensitive data.

3. **Integrated Evaluation Frameworks:** Combining synthetic data generators with comprehensive evaluation or attack frameworks can help explainability as well as ensuring an optimal balance between fidelity and privacy.

By addressing these challenges and focusing on these future directions, the practical application of synthetic data can be significantly enhanced, making it a more viable solution for real-world problems, particularly in sensitive domains such as healthcare.

7 Acknowledgements

1. This project was funded by Ecosystem Leadership Award under the EPSRC OobfJ22\100020.

2. This project was supported by the National Institute for Health and Care Research (NIHR)University College London Hospitals (UCLH) Biomedical Research Centre (BRC).

3. The authors thank Olajumoke Olatunji for her help in improving the documentation of SSG and helping us present it.

References

[1] Joshua Allen, Sarah Bird, and Kathleen Walker. Opendp platform for differential privacy, May 2020.

[2] Alhanoof Althnian, Duaa AlSaeed, Heyam Al-Baity, Amani Samha, Alanoud Bin Dris, Najla Alzakari, Afnan Abou Elwafa, and Heba Kurdi. Impact of dataset size on classification performance: An empirical evaluation in the medical domain. Applied Sciences, 11(2), 2021.

[3] Márcio Antunes and Ernesto Oliveira. Survey on synthetic data generation, evaluation methods and gans. Mathematics, 10(15):2733, 2022.

[4] M. Bayer. Sqlalchemy. `https://www.sqlalchemy.org/`.

[5] Emily E. Berkson, Jared D. VanCor, Steven Esposito, Gary Chern, and Mark D. Pritt. Synthetic data generation to mitigate the low/no-shot problem in machine learning. In 2019 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), pages 1–8. IEEE, 2019.

[6] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G. Willcocks. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. IEEE Trans. Pattern Anal. Mach. Intell., 44(11):7327–7347, 2022.

[7] Airbnb New User Bookings. Airbnb new user bookings. Kaggle, `https://www.kaggle.com/competitions/airbnb-recruiting-new-user-bookings`, 2015. [Accessed: November, 25, 2015].

[8] Kuntai Cai, Xiaokui Xiao, and Graham Cormode. Privlava: Synthesizing relational data with foreign keys under differential privacy. Proceedings of the ACM on Management of Data, 1:1–25, 06 2023.

[9] FK Dankar and K El Emam. Practicing differential privacy in health care: A review. Transactions on Data Privacy, 6(1):35–67, 2013.

[10] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci., 9(3-4):211–407, 2014.

[11] Joao Fonseca and Fernando Bação. Tabular and latent space synthetic data generation: a literature review. Journal of Big Data, 10, 07 2023.

[12] C. Gavidia-Calderon, M. Hauru, I. Stenson, and M. Yong. sqlsynthgen. `https://github.com/alan-turing-institute/sqlsynthgen`, 2024.

[13] Matt Hancock. Data saves lives: reshaping health and social care with data, 2022. Accessed: 2024-06-04.

[14] S. Harris, T. Bonnici, T. Keen, W. Lilaonitkul, M. J. White, and N. Swanepoel. Clinical deployment environments: Five pillars of translational machine learning for health. Frontiers in Digital Health, 4:939292, Aug 2022.

[15] Florimond Houssiau, James Jordon, Samuel N. Cohen, Owen Daniel, Andrew Elliott, James Geddes, Callum Mole, Camila Rangel-Smith, and Lukasz Szpruch. Tapas: a toolbox for adversarial privacy auditing of synthetic data, 2022.

[16] LA Jones, JR Nelder, JM Fryer, PH Alsop, MR Geary, M Prince, and RN Cardinal. Public opinion on sharing data from health services for clinical and research purposes without explicit consent: an anonymous online survey in the uk. BMJ Open, 12(4):e057579, Apr 2022.

[17] James Jordon, Lukasz Szpruch, Florimond Houssiau, Mirko Bottarelli, Giovanni Cherubin, Carsten Maple, Samuel N. Cohen, and Adrian Weller. Synthetic data - what, why and how? CoRR, abs/2205.03257, 2022.

[18] James Jordon, Lukasz Szpruch, Francois Houssiau, and Matteo Bottarelli. Synthetic data - what, why and how?, 2022.

[19] A Kopp. Microsoft smartnoise: Differential privacy machine learning case studies. Technical report, Microsoft, 2021. Accessed: 2024-06-08.

[20] Ryan McKenna, Brett Mullins, Daniel Sheldon, and Gerome Miklau. AIM: an adaptive and iterative mechanism for differentially private synthetic data. Proc. VLDB Endow., 15(11):2599–2612, 2022.

[21] J. P. Near and C. Abuah. Programming Differential Privacy, volume 1. 2021.

[22] Beata Nowok, Gillian M. Raab, and Chris Dibben. synthpop: Bespoke creation of synthetic data in r. Journal of Statistical Software, 74(11):1–26, 2016.

[23] Cian O'Donovan, Sonya Coleman, Dermot Kerr, Christian Cole, Simon Li, David Sarmiento Perez, and Hari Sood. Trusted research environment users. November 2023.

[24] Office for National Statistics. Ons methodology working paper series number 16: Synthetic data pilot. Technical report, Office for National Statistics, 2021.

[25] Observational Medical Outcomes Partnership (OMOP). Omop common data model. `https://www.ohdsi.org/data-standardization/the-common-data-model/`, 2024.

[26] A. Patra, R. Batra, A. Chandrasekaran, and C. Kim. A multi-fidelity information-fusion approach to machine learn and predict polymer bandgap. Computational Materials Science, 172:109280, 2020.

[27] C. Santoni, D. Zhang, Z. Zhang, and D. Samaras. Toward ultra-efficient high fidelity predictions of wind turbine wakes: Augmenting the accuracy of engineering models via les-trained machine learning. arXiv preprint arXiv:2404.07938, 2024.

[28] G. Schomerus et al. The stigma of alcohol-related liver disease and its impact on healthcare. Journal of Hepatology, 77(2):516–524, Aug 2022.

[29] Synth Team. Synth: The open source declarative data generator, 2021.

[30] A. Tucker, Z. Wang, Y. Rotalinti, and P. Myles. Generating high-fidelity synthetic patient data for assessing machine learning healthcare software. npj Digital Medicine, 3, 2020.

[31] West JD Vasan K. The hidden influence of communities in collaborative funding of clinical science. R Soc Open Sci., 8(8), 2021.

[32] Jason Walonoski, Mark Kramer, James Nichols, Marc Galdzicki, Amelia Quina, Christopher Moesel, Darrell Hall, Tim Duffield, Matthew Gratch, Pascal Coorevits, David Sundwall, Emily Grant, Colin Jones, and Liza Tong. Synthea: Synthetic patient population simulator, 2020. Accessed: 2024-06-08.

[33] Rachel M. Werner and David A. Asch. The unintended consequences of publicly reporting quality information. Journal of the American Medical Association, 293(10):1239–1244, 2005.

Differential Privacy for Time Series: A Survey

Yulian Mao[1,2], Qingqing Ye[2 *], Qi Wang[1,3], Haibo Hu[2]
[1]Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China
[2]Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University, Hong Kong SAR, China
[3]Shenzhen Key Laboratory of Safety and Security for Next Generation of Industrial Internet, Shenzhen, China
yulian.mao@connect.polyu.hk, qqing.ye@polyu.edu.hk, wangqi@sustech.edu.cn, haibo.hu@polyu.edu.hk

Abstract

Time series are extensively used in finance, healthcare, IoT, and smart cities. However, in many applications, time series often contain personal information, so releasing them publicly can pose privacy risks. Differential privacy has recently emerged as the state-of-the-art approach for safeguarding data privacy. Unfortunately, adapting differential privacy to time series presents unique challenges compared to other data types due to their large volume, temporal correlations, and dynamic nature. To address users' demands for time series analysis while simultaneously protecting privacy, a significant body of research works have been proposed. The aim of this survey is to summarize these works and provide a holistic view of the DP mechanisms under differential privacy. Furthermore, we will discuss the challenges associated with time series release, especially in one of its most prevalent applications — location based services. Finally, we will explore open challenges and shed light on directions for future research.

1 Introduction

Time series are being generated on a large scale across a wide range of application domains, such as IoT, finance, healthcare monitoring, operational event logs, and smart home sensors. For example, smart home devices such as thermostats and humidity sensors generate continuous time series on environmental conditions, tracking temperature fluctuations and moisture levels to optimize home climate control based on user activities. Additionally, trajectories represent a unique type of time series that contain both spatial and temporal information, such as GPS data tracking the movement of vehicles or individuals. To facilitate the analysis of time series and support various downstream tasks, numerous methods have been proposed, ranging from traditional statistical techniques, such as ARIMA [1] and exponential smoothing [2], to advanced machine learning models, such as long short-term memory (LSTM) networks [3]. However, a key issue in these time series applications is privacy. Since many data sources such as smart home sensors and location trajectories contain individuals' private information, the direct release or analysis of such time series can lead to significant privacy violations. Consequently, developing privacy-preserving mechanisms for time series analysis is essential.

Differential privacy (DP) [4] is a paradigm of privacy-preserving mechanisms that provides a theoretical privacy guarantee and has been further extended to the local setting to accommodate more general scenarios [5, 6].

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

* Dr. Qingqing Ye is the corresponding author.

Numerous DP mechanisms have been developed to support various queries [7, 8, 9], including estimating statistics such as frequency [7, 10] and mean [11], ensuring that the privacy of individuals in the dataset is protected even when aggregate information is released. Recently, research has shifted towards more complex applications, such as graph data mining [12] and machine learning problems [13]. Techniques such as differentially private stochastic gradient descent (DP-SGD) [14, 15] have been developed to train machine learning models with differential privacy guarantees, enabling the use of private datasets for tasks like classification and prediction without compromising individuals' privacy. It is worth noting that differential privacy is also being explored in the context of time series [16, 17, 19]. This involves developing new mechanisms capable of handling the features of time series, ensuring that privacy is maintained.

Nevertheless, time series data present more challenges compared to other types of data due to their large volume, temporal correlation, and dynamic nature. The large volume, in particular, poses significant issues for privacy models, as protecting every element of a time series would degrade utility. To address this challenge, three privacy levels have been proposed [19]: event-level privacy, which protects a single element in the time series; w-event level privacy, which provides a privacy guarantee for w consecutive elements; and user-level privacy, which protects all elements associated with an individual. To enhance utility, sampling and filtering-based mechanisms [18], as well as privacy budget allocation strategies [19], have been suggested. Additionally, the correlations between elements can lead to privacy breaches [20], necessitating countermeasures to confine these correlations [21, 22]. Given numerous research efforts in this field, a taxonomy is necessary to summarize the existing works and identify areas for future research.

However, to the best of our knowledge, there is no up-to-date and comprehensive survey specifically for time series under differential privacy. Dwork and Roth [23] coauthored a comprehensive survey on differential privacy, which seems outdated now in terms of state-of-the-art techniques. More recently, there are two surveys from Zhao et al. [24, 25] focusing on the concepts and applications of differential privacy, but they do not extensively cover time series. Miranda-Pascual et al. [26] conducted a survey on trajectory data publication, which mainly talks about downstream tasks with little emphasis on privacy preservation. The most relevant survey on time series under differential privacy is by Katsomallos et al. [27]. However, since it was published in 2019, the paper does not reflect current technical trends, such as LDP, which now accounts for a crucial portion of privacy-preserving time series research.

In this survey, we aim to provide a comprehensive review of research works on time series under differential privacy. The main contents and paper organization are summarized as follows.

- **Section 2: Fundamental concepts of time series and differential privacy.** We first introduce the basic concepts of time series and differential privacy, including the definitions of differential privacy and the composition theorems. Additionally, we elucidate the three privacy levels specifically defined within the context of time series.

- **Section 3: Count queries and corresponding advanced queries.** We begin with an introduction to count queries, and then present two core techniques for their realization: the binary tree-based mechanism [17] and the matrix mechanism [28]. Following this, we discuss advanced queries, such as frequency and histogram estimations, which are based on count queries. Finally, we explore downstream applications derived from count queries.

- **Section 4: Sum/mean queries and downstream applications.** We list sum and mean queries together due to their inherent correlation. Following the introduction of sum and mean queries, we present the developed mechanisms for these queries. Subsequently, we review the literature on downstream applications.

- **Section 5: Time series release.** We classify the literature into two categories: methods based on value perturbation and methods based on synthesis. For value perturbation-based methods, we first review privacy budget allocation strategies and then present the optimization strategies to improve utility. We then introduce the synthesis-based methods, including those based on statistics and generative models. Additionally, we discuss the privacy models of the time series mechanisms and review a line of work that perturbs the temporal order rather than the values to accommodate value-critical scenarios.

- **Section 6: Location based services and trajectory release.** Given that location based services are common applications under differential privacy, we dedicate a section to discussing the relevant literature. To improve the utility, geo-indistinguishability [29] was proposed to constrain the perturbation domain. Moreover, due to the apparent temporal correlation, the relationships between locations need to be considered. Finally, we present mechanisms designed for trajectory release based on perturbation and synthesis.

- **Section 7: Open challenges.** We present a few future research directions for DP-based time series in terms of privacy model, potential correlation-based attacks, complex data type, and learning based problems.

2 Preliminaries

In this section, we first introduce the basic concepts of differential privacy and differential privacy for time series. As for the latter, we mainly focus on different privacy levels of DP mechanisms used to analyze time series.

2.1 Differential Privacy

Differential privacy is a rigorous and practical formalization that provides a quantitative measure of privacy leakage for an individual when participating in a database [4]. In the nearly 20 years since its inception, differential privacy has become widely adopted as a privacy-preserving framework. Many companies, such as Microsoft [30], Google [31], and Apple [32], utilize differential privacy to collect users' data while providing privacy guarantees. Additionally, the US Census Bureau adopted differential privacy for the 2020 decennial census [33].

Based on utilization scenarios, differential privacy can be broadly categorized into centralized differential privacy (CDP) and local differential privacy (LDP) [6]. Centralized differential privacy requires a trusted third party to act as the data curator, collecting data from users and releasing the processed results to the public. The trusted third party is assumed to safeguard private information and not disclose it. However, in many situations, such a trusted third party may not exist. Consequently, local differential privacy has been proposed, allowing data to be sanitized locally before being uploaded. These two different scenarios are depicted in Fig 1, and their formal definitions are provided below.

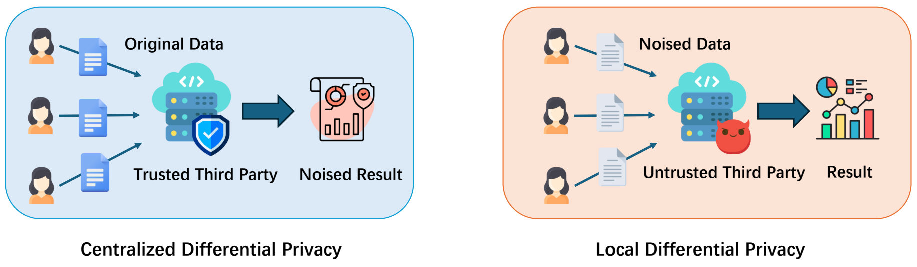


Figure 1: An illustration for centralized differential privacy and local differential privacy. In the context of centralized differential privacy, a trusted third party collects original data from users and adds noise to the processed result. In contrast, under local differential privacy, users add noise to their data locally before uploading the noised data to the untrusted third party.

2.1.1 Centralized Differential Privacy (CDP)

Before delving into the formal definition of centralized differential privacy, it is important to first elucidate some underlying concepts. We start with the definition of neighboring datasets.

Definition 2.1: *[Neighboring Datasets [23, 24, 34]] Two datasets D and D' are neighboring if they only differ by only one record. In Unbounded CDP, D can be obtained from D' by adding or removing one record, whereas in Bounded DP, D can be obtained from D' by replacing one record.*

Definition 2.2: *[(ϵ, δ)-Centralized Differential Privacy ((ϵ, δ)-CDP) [23, 24, 34]] A randomized mechanism $\mathcal{M}$ satisfies (ϵ, δ)-centralized differential privacy if and only if for any two neighboring datasets D, D', and any possible output $R \subseteq \mathrm{Range}(\mathcal{M})$, there is*

$$\Pr[\mathcal{M}(D) = R] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(D') = R] + \delta.$$

When $\delta = 0$, $\mathcal{M}$ satisfies ϵ-centralized differential privacy.

2.1.2 Local Differential Privacy (LDP)

As we aforementioned, local differential privacy is adopted in a local mode. Compared with CDP, LDP requires more added noise to ensure privacy but does not need a trusted third party. Therefore, neighboring datasets in LDP can be any input from users.

Definition 2.3: *[(ϵ, δ)-local differential privacy ((ϵ, δ)-LDP) [24, 34]] A randomized mechanism $\mathcal{A}$ satisfies (ϵ, δ)-local differential privacy if and only if for any inputs v, v', and any possible output $r \subseteq \mathrm{Range}(\mathcal{A})$, there is*

$$\Pr[\mathcal{A}(v) = r] \leq e^{\epsilon} \cdot \Pr[\mathcal{A}(v') = r] + \delta.$$

When $\delta = 0$, $\mathcal{A}$ satisfies ϵ-local differential privacy which is also called pure-LDP [7].

2.1.3 Composition Theorems

Under differential privacy (both CDP and LDP), there are two useful composition theorems [34]: sequential composition and parallel composition.

Definition 2.4: *[Sequential Composition [34]] Given a dataset x, and two mechanisms M_1, M_2 satisfy (ϵ_1, δ_1)-DP and (ϵ_2, δ_2)-DP, respectively, the mechanism $M = (M_1(x), M_2(x))$ satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DP.*

Definition 2.5: *[Parallel Composition [34]] Given a mechanism M satisfy (ϵ, δ)-DP, and the k disjoint separations of the dataset x (i.e., $x_1 \cup x_2 \cup \cdots \cup x_k = x$), the release $M(x_1), M(x_2), \cdots, M(x_k)$ satisfies (ϵ, δ)-DP.*

2.1.4 Pufferfish Privacy

When providing privacy guarantees for correlated time series, differential privacy faces the challenge of excessive noise addition. Specifically, group differential privacy [23] necessitates adding $O(T)$ noise for a correlated time series with length T, leading to significant utility degradation. To address correlated data, pufferfish privacy, a generalized version of differential privacy, was proposed [35]. In addition to the privacy budget ϵ, pufferfish privacy requires three additional parameters [36]: a set of secrets $\mathcal{S}$ representing users' private data, a set of secret pairs $\mathcal{Q} \subseteq \mathcal{S} \times \mathcal{S}$ that must remain indistinguishable, and a class of data distribution Θ indicating the correlation.

Definition 2.6: *[ϵ-Pufferfish Privacy [36]] Give the parameters $\mathcal{S}$, $\mathcal{Q}$, and Θ, a randomized mechanism M satisfies ϵ-pufferfish privacy if $\forall \theta \in \Theta$ with $X \sim \theta$, $\forall (s_i, s_j) \in \mathcal{Q}$, $\forall w \in Range(M)$, there is*

$$\Pr(M(X) = w | s_i, \theta) \leq e^{\epsilon} \cdot \Pr(M(X) = w | s_j, \theta),$$

when $\Pr[\mathrm{s_i}|\theta] \neq 0$ and $\Pr[\mathrm{s_j}|\theta] \neq 0$.

2.1.5 Differential Privacy Mechanisms

Given that noise addition from a distribution is a fundamental implementation of differential privacy, this technique is extensively used for statistical estimation. Numerous applications rely on count queries, including frequency estimation, histogram estimation, and top-k item mining. Korolova et al. [37] introduced a mechanism for releasing search log histograms by adding Laplace noise. Xiao et al. [38] proposed a wavelet-based mechanism to release range count queries. In the context of LDP, Wang et al. [7] reviewed existing mechanisms for frequency estimation and introduced two optimized approaches. Li et al. [11] developed a mechanism for estimating numerical data distributions. Wang et al. [9] proposed a prefix extension mechanism to identify the top-k frequent items within a large domain. Beyond count queries, many applications focus on sum or mean queries. Wang et al. [39] devised a mechanism with optimized perturbation probability for numerical data under LDP. Xue et al. [40] introduced a mean estimation mechanism that supports personalized privacy budgets for each user. Zhou et al. [41] developed a mechanism to estimate the mean of sparse vectors. To incorporate data knowledge, Wei et al. [42] proposed an optimized mean estimation mechanism based on data distribution estimated in the initial phase under LDP. Beyond statistical estimations, data release for downstream tasks is another critical application. Ye et al. [43] proposed a mechanism to release edge information for clustering coefficient estimation. Ma et al. [44] developed a mechanism to construct decision trees under LDP, enhancing utility through the adoption of public data. Additionally, since the introduction of Differentially Private Stochastic Gradient Descent (DP-SGD) [14], numerous privacy-preserving learning-based mechanisms have been proposed under differential privacy.

Across the various scenarios, time series represent a unique research field due to their sequential nature and temporal dependencies. This adds complexity to ensuring differential privacy while maintaining data utility. In the following sections, we will introduce the concepts of time series under differential privacy.

2.2 Differential Privacy for Time Series

In this subsection, we will introduce the concept of time series and the specific definitions of differential privacy for time series, including various privacy levels. Additionally, we will provide a concise roadmap of this survey.

2.2.1 Time Series

In general, a time series is regarded an ordered sequence of values with finite length [45], while data streams are continuously generated sequences with infinite length [46]. For ease of presentation, both are referred to as “time series" in this paper, encompassing both finite and infinite settings.

Definition 2.7: *[Time Series [45, 46, 47]] A time series S is an ordered sequences of values, i.e., $S = \{S_{t_1}, S_{t_2}, S_{t_3}, \cdots\}$. For simplicity, the timestamp is usually omitted, and a time series is denoted as $S = \{S_1, S_2, S_3, \cdots\}$. If any element $S_i \in \mathbb{R}$, the time series is called a univariate infinite time series. Otherwise, the time series is a multivariate infinite time series if $S_i \in \mathbb{R}^d$, namely, each element is with d dimensions.*

Note that if a time series has a finite length, it is called a finite time series. Otherwise, it is referred to as an infinite time series.

2.2.2 Privacy Levels

In the context of time series, three major privacy levels have been proposed based on the privacy guarantees. Event-level privacy only protects a single element within a time series, w-event level privacy provides a privacy guarantee for a sequence of w consecutive elements, and user-level privacy protects the entire time series. The corresponding definitions are provided below, with illustrations depicted in Fig. 2.

Definition 2.8: *[Event-Level Adjacent Time Series [19]] For two time series S and S', they are event-level adjacent if*

1) *There exists a timestamp i, $S_i \neq S'_i$;*

2) *For any other timestamp j, $S_j = S'_j$.*

Definition 2.9: *[Event-Level Privacy [19]] A randomized mechanism $\mathcal{M}$ satisfies (ϵ, δ)-event-level differential privacy if and only if for any two event-level adjacent time series S, S', and any possible output $R \subseteq \mathrm{Range}(\mathcal{M})$, there is*

$$\Pr[\mathcal{M}(S) = R] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(S') = R] + \delta.$$

When $\delta = 0$, $\mathcal{M}$ satisfies ϵ-event-level differential privacy.

Definition 2.10: *[w-Event Level Adjacent Time Series [19]] For two time series S and S', they are w-event level adjacent if*

1) *There exists w consecutive timestamp $\{k, k+1, \cdots, k+w-1\}$ and for any $i \in \{k, k+1, \cdots, k+w-1\}$, $S_i \neq S'_i$;*

2) *For any other timestamp j, $S_j = S'_j$.*

Definition 2.11: *[w-Event Level Privacy] A randomized mechanism $\mathcal{M}$ satisfies (ϵ, δ)-w-event level differential privacy if and only if for any two w-event level adjacent time series S, S', and any possible output $R \subseteq \mathrm{Range}(\mathcal{M})$, there is*

$$\Pr[\mathcal{M}(S) = R] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(S') = R] + \delta.$$

When $\delta = 0$, $\mathcal{M}$ satisfies ϵ-w-event level differential privacy.

Definition 2.12: *[User-Level Adjacent Time Series] For two time series S and S', they are user-level adjacent if for all timestamps $t_{u_i} = \{t_1, t_2, \cdots, t_k\}$ from any user u_i, there is $S_j \neq S'_j, \forall j \in t_{u_i}$. Note that t_{u_i} can be infinite for infinite time series.*

Definition 2.13: *[User-Level Privacy] A randomized mechanism $\mathcal{M}$ satisfies (ϵ, δ)-user-level differential privacy if and only if for any two user-level adjacent time series S, S', and any possible output $R \subseteq \mathrm{Range}(\mathcal{M})$, there is*

$$\Pr[\mathcal{M}(S) = R] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(S') = R] + \delta.$$

When $\delta = 0$, $\mathcal{M}$ satisfies ϵ-user-level differential privacy.

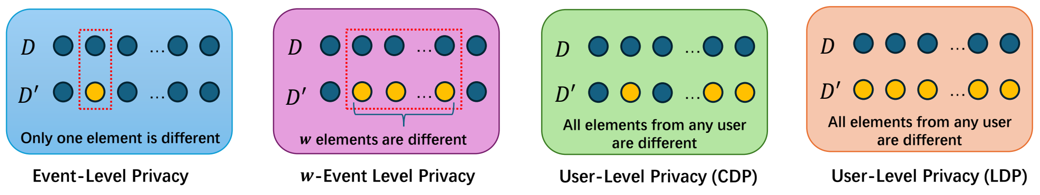


Figure 2: An illustration for privacy levels. In terms of event-level privacy, there is only one element different in the neighboring datasets. While for the w-event level privacy, there are w consecutive elements that differ in the neighboring datasets. For user-level privacy, all the elements from any user can be different in the neighboring datasets. Note that under LDP, the two user-level adjacent time series are from different users, which means the elements could be entirely different.

Obviously, event-level privacy provides the lowest level of privacy but requires the least amount of noise. Conversely, user-level privacy guarantees the strongest privacy but necessitates the largest amount of noise, which can significantly degrade utility.

2.3 Roadmap of This Survey

Since count queries and sum/mean queries are fundamental statistical operations, many advanced queries and downstream applications are derived from them. This survey begins with a review on count queries, followed by a discussion of sum/mean queries. Each subsection on these queries starts with an introduction to the concepts, followed by a review of their downstream applications. Subsequently, we introduce data release mechanisms designed to publicize data for downstream tasks. Given the popularity of location based services in time series applications, we dedicate a separate section to trajectories, reviewing the literature on location perturbation, temporal correlation issues, and trajectory release. To suggest future directions, we propose open challenges related to privacy models, temporal correlation-based attacks, complex data types, and learning-based problems. The roadmap for this survey is illustrated in Fig. 3.

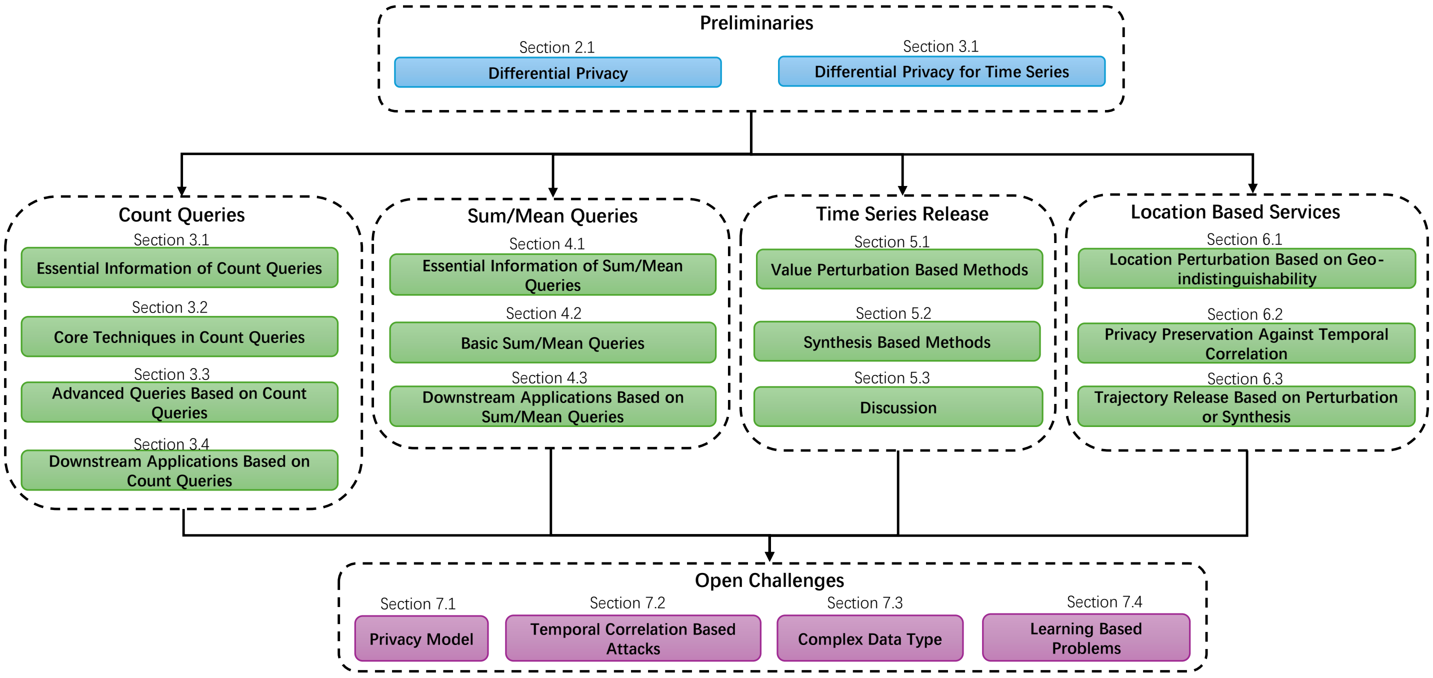

Preliminaries
Section 2.1
Differential Privacy
Section 3.1
Differential Privacy for Time Series
Count Queries
Section 3.1
Essential Information of Count Queries
Section 3.2
Core Techniques in Count Queries
Section 3.3
Advanced Queries Based on Count Queries
Section 3.4
Downstream Applications Based on Count Queries
Sum/Mean Queries
Section 4.1
Essential Information of Sum/Mean Queries
Section 4.2
Basic Sum/Mean Queries
Section 4.3
Downstream Applications Based on Sum/Mean Queries
Time Series Release
Section 5.1
Value Perturbation Based Methods
Section 5.2
Synthesis Based Methods
Section 5.3
Discussion
Location Based Services
Section 6.1
Location Perturbation Based on Geo-indistinguishability
Section 6.2
Privacy Preservation Against Temporal Correlation
Section 6.3
Trajectory Release Based on Perturbation or Synthesis
Open Challenges
Section 7.1
Privacy Model
Section 7.2
Temporal Correlation Based Attacks
Section 7.3
Complex Data Type
Section 7.4
Learning Based Problems

Figure 3: Roadmap of this survey.

3 Differential Privacy for Count Queries

In this section, we introduce the concept of count queries within the framework of differential privacy, a topic that has garnered substantial research attention.

3.1 Essential Information of Count Queries

For a time series S comprising categorical values with length t (t can be ∞), a count query can be formally expressed as follow:

$$\mathrm{F}_{\text{cnt}}(v, S) = \sum_{i=1}^{t} \mathbf{1}_v(x_i),$$

where $\mathbf{1}_v(x_i)$ denotes the indicator function, defined below:

$$\mathbf{1}_v(x_i) := \begin{cases} 1 & \text{if } x_i = v, \\ 0 & \text{if } x_i \neq v. \end{cases}$$

Count queries are fundamental in the context of differential privacy, as they form the basis for other tasks, such as frequency and histogram estimations. Compared with other scenarios, applying count queries to time series presents unique challenges. Time series arrives as a continuous stream, necessitating the injection of a larger amount of noise to provide the desired privacy guarantees due to continuous release of query results.

3.2 Core Techniques in Count Queries

For continuously releasing a finite time series, a naive approach is to release each element with the entire privacy budget ϵ [17]. However, such a method would introduce substantial additive noise, leading to low utility with an error bound of $O(\frac{\sqrt{T}}{\epsilon})$. Dwork et al. [16] proposed the first work to handle binary time series under event-level CDP with a logarithm error bound. However, this mechanism is limited to time series with finite length T. Subsequently, Chan et al. [17] improved the mechanism to support the release of infinite binary time series. Both of these works employ a tree-based method to enhance utility. The binary tree mechanism [17], illustrated in Fig. 4, ensures that each release influences at most one node at each level for finite time series. Consequently, each node only needs to add noise corresponding to the privacy budget $\frac{\epsilon}{(\log T+1)}$. To guarantee a logarithm error bound for infinite time series [17, 48], more binary trees will be construed. Since the update of one element only influences one tree, each tree will be allocated an entire privacy budget. More details can be referred to Fig. 4.

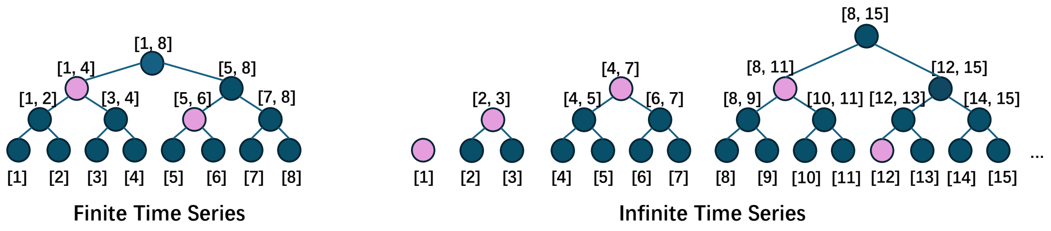


Figure 4: Here is an illustration of binary tree mechanism for time series count query under event-level CDP. For finite time series $[1, 8]$ (i.e., $T = 8$), the count query result for the range $[1, 6]$ is specified as $\hat{\mathrm{F}}_{\mathrm{cnt}}(v, [1, 6]) = \mathrm{F}_{\mathrm{cnt}}(v, [1, 4]) + \mathrm{F}_{\mathrm{cnt}}(v, [5, 6]) + 2Lap(\frac{\log T+1}{\epsilon})$, where $Lap(\cdot)$ is drawn from the Laplace distribution with zero mean. For an infinite time series, multiple binary trees will be constructed. Since a change in any single element only influences one tree, each tree will be allocated an entire privacy budget. The overall privacy budget consumption of the mechanism is ϵ, which can be calculated using parallel composition [34]. For example, $\hat{\mathrm{F}}_{\mathrm{cnt}}(v, [1, 12]) = \hat{\mathrm{F}}_{\mathrm{cnt}}(v, [1, 1]) + \hat{\mathrm{F}}_{\mathrm{cnt}}(v, [2, 3]) + \hat{\mathrm{F}}_{\mathrm{cnt}}(v, [4, 7]) + \hat{\mathrm{F}}_{\mathrm{cnt}}(v, [8, 11]) + \hat{\mathrm{F}}_{\mathrm{cnt}}(v, [12, 12])$.

In addition to the tree-based structure, another approach to handle time series under differential privacy is based on the matrix mechanism [28, 49]. Without privacy concerns, a count query $\mathcal{M}$ for binary time series x with length n can be specified as

$$\mathcal{M}(x) = Mx = \begin{bmatrix} 1 & 0 & 0 & \cdots \\ 1 & 1 & 0 & \cdots \\ 1 & 1 & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} x = \begin{bmatrix} x_1 \\ x_1 + x_2 \\ \vdots \\ \sum_1^n x_i \end{bmatrix}.$$

Based on this formulation, the matrix mechanism is employed to release data streams under (ϵ, δ)-DP for further error reduction. Given a workload matrix M, the strategy matrix R and reconstruction matrix L are first constructed, denoted as $M = LR$. The strategy matrix R is utilized to pre-process the input x. After adding a Gaussian noise vector z to the processed term Rx, a post-processing step L is applied. In summary, for an input time series $x \in \mathbb{R}^n$, the matrix mechanism is denoted as

$$\mathcal{M}_{L,R}(x) = L(Rx + z),$$

where $z \sim N(0, \|R\|^2_{1\to 2} C^2_{\epsilon,\delta}\mathbf{I})$ to ensure (ϵ, δ)-DP, $\|R\|^2_{1\to 2}$ is the maximum of the 2-norm of the columns of the strategy matrix R. The additive mean squared error $\mathrm{err}_{\ell_2^2}$ of the matrix mechanism is

$$\mathrm{err}_{\ell_2^2}(\mathcal{M}_{L,R}, \mathcal{M}, n) = \max_{x\in\mathbb{R}^n} \mathbb{E}\left[\frac{1}{n}\left\|\mathcal{M}_{L,R}(x) - \mathcal{M}(x)\right\|_2^2\right] = \frac{1}{n} trace(L^T L)\|R\|^2_{1\to 2} C^2_{\epsilon,\delta}.$$

Hence, the matrix mechanism can be regarded as an optimization problem. For more comprehensive information, we recommend reading the work by Henzinger et al. [49].

3.3 Advanced Queries Based on Count Queries

Query	Ref.	Type	Privacy Level	Method Primitive	Error Bound
Binary Counting	[16]	finite	event-level CDP	binary tree mechanism	$O(\frac{1}{\epsilon} \cdot (\log^{1.5} t))$
	[17]	infinite	event-level CDP	binary tree mechanism	$O(\frac{1}{\epsilon} \cdot (\log^{1.5} t))$
	[48]	infinite	user-level CDP	binary tree mechanism	$O(\frac{\kappa(D_t)}{\epsilon} \cdot \log^{1.5} t \cdot \log^{1+\theta}(\kappa(D_t)))$
	[49]	infinite	event-level (ϵ, δ)-CDP	matrix mechanism	$O(\frac{4}{\epsilon^2}(\frac{4}{9} + \ln(\frac{1}{\delta}\sqrt{\frac{2}{\pi}}))(1 + \frac{\ln(4n/5))}{\pi})^2)$
Frequency Estimation	[50]	finite	event-level zCDP	multi-branch tree	$O(\tau \log T \sqrt{2(s+1)(t-1)\log(6T/\beta)})$
	[48]	infinite	user-level CDP	binary tree mechanism	$O(\frac{\kappa(D_t)}{\epsilon\theta} \cdot \log^{1+\theta}(\kappa(D_t))) \cdot \log(tR/\beta)$
Distinct Elements Counting	[51]	finite	user-level CDP	bipartite maximum matching	$O(\frac{\ell_*}{\epsilon} \log(\frac{\ell_{\max}}{\beta})$

Table 1: A brief summary of count-based queries, where t indicates the current timestamp, T is the length of the time series, D represents the dataset, $\kappa(D_t)$ denotes the maximum number of elements contributed by any user, τ is the privacy level of zero-concentrated differential privacy (zCDP) [52], θ is any small constant, ℓ is the sensitivity (ℓ_* represents the bounded sensitivity), and β is the confidence parameter. [1]

Based on basic binary count queries, numerous other types of count queries have been proposed, including frequency estimation, frequency moment estimation, and distinct element counting. A brief summary of the literature is presented in Table 1. Cardoso et al. [50] introduced differentially private histograms in the continual observation model with an unknown domain. To facilitate practical implementation, the authors propose a mechanism that continually returns a noisy histogram by aggregating counts at each round and adding noise to them. Dong et al. [48] proposed a mechanism to estimate frequency under user-level CDP. Since a user may contribute multiple elements to the time series, the mechanism first estimates each user's contribution and then applies a truncation process to retain only a limited number of elements per user, marking subsequent items as invalid. Furthermore, the proposed approach reduces the domain of elements to further enhance the utility of frequency estimates. The work by [51] proposes a method to estimate the number of distinct elements in a time series, and obtain the bounds on the true number of unique elements. The paper models the dataset as a bipartite graph and reduces the unique counting process to a max-flow problem, allowing the utilization of standard algorithms for bipartite maximum matching to solve unique counting problem. Furthermore, Kalemaj et al. [53] proposed a mechanism that achieves a logarithmic error bound for releasing distinct elements with insertions and deletions in a finite time series, under item-level differential privacy, which considers neighboring datasets

[1] The error bound of [49] in the table is the L_2 norm error.

differing by more than one element deletion. Epasto et al. [54] presented the first work to release differentially private ℓ_p frequency moments, denoted as $\sum_i f_i^p$. Notably, when $p = 1$, the frequency moment reduces to distinct counting. Practically, Zhang et al. [55] propose DP-SQLP, the first differentially private stream aggregation processing system, which has been implemented for Google Shopping and is planned for future application to Google Trends.

3.4 Downstream Applications Based on Count Queries

The application of differential privacy in count queries primarily involves releasing histograms and monitoring anomalies. Recent research [56] has demonstrated that employing subsampling and filtering techniques can reduce the sensitivity of real-time series, thereby enhancing the utility of differentially private mechanisms applied to such data. To improve utility, these mechanisms [18, 57, 58, 59] typically employ sampling to reduce the number of elements needing protection and filtering to mitigate the impact of added noise. Additionally, some mechanisms leverage pufferfish privacy [60, 61], which introduces less noise while achieving similar privacy guarantees. The details of these mechanisms are as follows. Fan et al. [18] introduced a mechanism for collecting count query results under the FAST framework, which employs filtering and adaptive sampling techniques to satisfy CDP. In a separate work, Fan et al. [57] applied the FAST framework for anomaly detection, specifically for detecting epidemic outbreaks. Li et al. [58] proposed a mechanism that only releases the histogram when it significantly differs from previous values, with the threshold adjusted according to feedback from the control system. Wang et al. [59] proposed a framework called SecWeb, following the idea of FAST under w-event level differential privacy. In addition to adaptive sampling and filtering, SecWeb incorporates dynamic grouping and injects Laplace noise based on the groups rather than individual elements. Wang et al. [8] proposed the first mechanism to achieve almost local differential privacy (LDP) under the w-event privacy model. Their approach involves employing multiple agents to collect data from users and release sanitized data to an untrusted third party. Liang et al. [60] introduced a mechanism for releasing web browsing histograms under the pufferfish privacy framework, which is beneficial for perturbing correlated data. Their proposed mechanism includes a model to quantify privacy leakage arising from temporal correlations and presents three strategies to enhance the model's efficiency: bounding the number of secret pairs, limiting the session length, and avoiding repetitive computations. Ding et al. [61] proposed a mechanism within the framework of pufferfish privacy to make the time and occurrence of an element indistinguishable.

Based on count queries, mechanisms proposed under the local differential privacy (LDP) framework are commonly used to estimate statistics. To conserve the privacy budget under LDP, a memoization technique [62] was proposed, which stores sanitized versions of all values for further release. [63] improved upon memoization by incorporating hashing. However, memoization may leak privacy in the presence of a knowledgeable adversary who can potentially derive changes without prior knowledge. Xue et al. [64] introduced a difference tree-based mechanism that applies fresh perturbation at each timestamp under user-level privacy, enabling the aggregation of statistics without violating changing points. Additionally, [65] proposed a method to reduce the item domain, thereby enhancing utility. Beyond memoization based methods, He et al. [66] proposed a privacy budget allocation strategy to enhance the utility of frequency release under w-event level condensed local differential privacy (CLDP) [67]. In their approach, the allocated privacy budget depends on the predicted elements, determined via a proportional-integral-derivative (PID) controller. In other applications, Feng et al. [68] proposed a mechanism to estimate the distribution of infinite time series while satisfying user-level differential privacy. Their approach achieves reasonable utility by bounding privacy leakage and optimizing the allocation of the privacy budget. Li et al. [69] introduced the first work on collecting the top-k items from a time series while satisfying event-level local differential privacy and adhering to a bounded memory space constraint. Their proposed mechanisms are based on the HeavyGuardian data structure, which maintains the frequently occurring elements while evicting the infrequent ones. Additionally, Gu et al. [70] introduced a mechanism under pattern-level privacy, which is similar to w-event level privacy but does not require successions. To privately release critical patterns (i.e., subsequences of elements in a time series), their mechanism perturbs the existence of each element, thereby providing a privacy guarantee.

4 Differential Privacy for Sum/Mean Queries

This section provides a comprehensive review of the literature concerning sum and mean queries within the differential privacy framework. It explores the concept of sum/mean queries, the basic queries, and the downstream applications.

4.1 Essential Information of Sum/Mean Queries

While preserving the occurrence of an element in a time series is important, maintaining the accuracy of the element's value is equally crucial. To ensure precise results for queries such as sum or mean, small deviations in the perturbation process are necessary. Since the mean is directly correlated to the sum, we discuss sum and mean queries together.

The sum query can be denoted as

$$\mathrm{F}_{sum}(T,S)=\sum_{i=1}^{T}S_i,$$

where T represents the timestamp for sum release and S is the corresponding time Series. The range query on sum is

$$\mathrm{F}_{rsum}((T_1,T_2),S)=\sum_{i=T_1}^{T_2}S_i,$$

where (T_1,T_2) represents the query range, and S is the corresponding time Series.

As for the mean query, it can be denoted as

$$\mathrm{F}_{mean}(T,S)=\frac{1}{T}\sum_{i=1}^{T}S_i,$$

where T represents the timestamp for mean release, and S is the corresponding time Series. Another common mean query is to release the mean at a timestamp from users' time series,

$$\mathrm{F}_{rtm}(t,S)=\frac{1}{n}\sum_{i=1}^{n}S_t^{u_i},$$

where $S_t^{u_i}$ represents the value at timestamp t from the user u_i, and n is the number of users.

4.2 Basic Sum/Mean Queries

There have proposed a line of work to release sum/mean queries under differential privacy, with a brief summary provided in Table 2. The pioneering work by Bolot et al. [71] was the first to study the continual decaying sums problem. They explored three variants: the window sum (range sum query), which releases the sum of W consecutive elements; the exponential decay sum, which releases the sum of elements weighted by an exponential function; and the polynomial sum, which releases the sum of elements weighted by a polynomial function. Henzinger et al. [72] also investigated the continual decaying sum problem. Their work introduced the use of the Gaussian mechanism for adding noise and derived tighter error bounds. In contrast, Dong et al. [48]

[2]The error bound of [72] in the table is the L_2 norm error.

Query	Ref.	Type	Privacy Level	Method Primitive	Error Bound
Sum	[48]	infinite	user-level CDP	binary tree mechanism	$O(\frac{\varphi(D_t)}{\epsilon\theta} \cdot \log^{1.5}(tR) \cdot \log^{1+\theta}(\varphi(D_t)) \cdot \log(1/\beta))$
Window Sum	[71]	finite	event-level CDP	binary tree mechanism	$O(\frac{1}{\epsilon} \log W \frac{1}{\beta})$
	[72]	finite	event-level (ϵ, δ)-CDP	matrix mechanism	$O(2\sigma_{\epsilon,\delta}^2 \Delta^2 (1 + \frac{\log W}{\pi} + \frac{2}{W})^2)$
Exponential Decay Sum	[71]	finite	event-level CDP	binary tree mechanism	$O(\frac{1}{\epsilon} \log \frac{\alpha}{1-\alpha} \frac{1}{\beta})$
	[72]	finite	event-level (ϵ, δ)-CDP	matrix mechanism	$O(\sigma_{\epsilon,\delta}^2 \Delta^2 (1 + \frac{1}{\pi} S_{T,2\alpha})^2)$
Polynomial Decay Sum	[71]	finite	event-level CDP	binary tree mechanism	$\Omega(1 - \frac{\epsilon^{c-1}}{\log^{c-1}(1/\beta)})$
	[72]	finite	event-level (ϵ, δ)-CDP	matrix mechanism	$O(\sigma_{\epsilon,\delta}^2 \Delta^2 (1 + \frac{H_{T,2c-1}}{4})^2)$

Table 2: The table provides a brief summary of sum-based queries, where $\varphi(D_t)$ denotes the maximum contribution from any user at time t, θ is a small constant, W is the window length, β is the confidence parameter, α indicates the exponential decay parameter, c represents the polynomial decay parameter, $H_{T,2c}$ is the generalized Harmonic sum, $S_{T,2\alpha}$ is a defined series sum with $\alpha > 1$.[2]

addressed sum queries by reducing them to count queries, as their approach could only handle the latter. For each timestamp with value x_i, they expand it into R steps, where the first $x_i (w.l.o.g., x_i < R)$ steps are filled with 1, and the remaining steps are filled with a special symbol $\perp$. However, this method requires prior knowledge of the maximum value R that can occur in the time series. [73] proposed a method for answering sum queries with a threshold under a multi-branch tree structure. The threshold is optimized based on the expected squared error between the true result and the estimated one. Their proposed mechanism cannot handle infinite time series, so they claim that most queries focus on a limited range, allowing for truncation of the infinite time series. Additionally, their work introduced the first mechanism to release time series under LDP with a threshold for value truncation. Instead of directly perturbing the values, the mechanisms proposed by Ye et al. [74, 75] perturb the temporal order. This approach makes them naturally adaptable for sum/mean queries while preserving the original values. These methods demonstrate superior performance for calculating moving averages.

4.3 Downstream Applications Based on Sum/Mean Queries

Due to utility considerations, existing mechanisms for downstream applications based on sum and mean queries primarily operate under event-level privacy or w-event level privacy. Perrier et al. [76] introduced a differentially private mechanism for publishing statistics of real-valued time series under event-level privacy, such as moving averages derived from energy data collected through smart meters. Their approach addresses scenarios where the bound on observations is either overly conservative or unknown, which is crucial for real-time monitoring applications. The proposed mechanism optimizes utility by scaling the added noise to the threshold value instead of a potentially larger bound, thereby improving accuracy. To enable real-time computation of the mean at any timestamp from users' time series under w-event LDP, Wang et al. [77] proposed sampling strategy to select important elements and a privacy budget allocation strategy according to the importance of the elements. However, the sampling process may inadvertently reveal some private information due to the intentional selection. Kurt et al. [78] proposed an online anomaly detection method for networks based on the cumulative sum algorithm, satisfying event-level (ϵ, δ)-differential privacy. Their approach adds noise to the statistic at each timestamp from each network node, operating under event-level privacy, and then derives the mean from the data of all nodes. This allows for detecting anomalies by monitoring changes in the released means.

5 Differential Privacy for Time Series Release

Time series release aims to publicly share private time series while preserving privacy. Value perturbation methods add noise to the data values, often using sampling and adaptive budget allocation for utility. While temporal perturbation methods dispatch elements across timestamps to obfuscate event timings, avoiding value distortion but risking empty releases or collisions.

5.1 Value Perturbation Based Methods

Time series release aims to directly publicize the time series for downstream tasks. Since time series release focuses on preserving the accuracy of values, privacy budget allocation is critical for controlling added noise and minimizing distortion. Therefore, a sampling-based method is often employed to select crucial elements according to the tasks, reducing the number of points requiring privacy budget allocation and enhancing utility. Additionally, to further improve the utility of the released data, a post-processing step can be adopted to correct the noisy data using prior knowledge. The outline of time series release is summarized in Fig. 5.

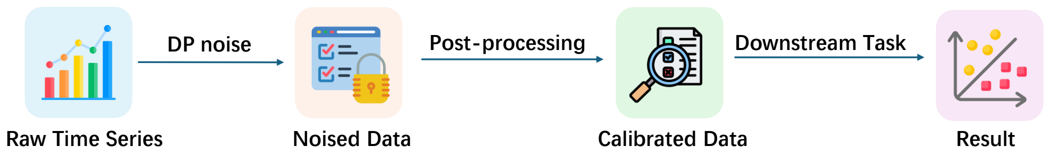


Figure 5: The outline of time series release under differential privacy.

5.1.1 Privacy Budget Allocation Strategies

User-level privacy protects the elements from any user but requires a larger privacy budget for reasonable utility, particularly challenging for time series release. Conversely, event-level privacy safeguards individual elements, yet may not suffice for comprehensive privacy guarantees. To balance the privacy levels, Kellaris et al. [19] first proposed w-event level privacy under CDP. By leveraging w-event level privacy, sanitized time series can offer better privacy guarantees than event-level privacy and higher utility than user-level privacy. To optimize the advantages of w-event level privacy, Kellaris et al. [19] proposed two privacy budget allocation strategies. Building upon the ideas of the budget distribution and absorption strategies in [19], Ren et al. [79] proposed corresponding strategies under LDP framework. To mitigate the utility degradation caused by dividing the privacy budget, the authors divide the users instead, with each user reporting only one timestamp within a w-length window. Several sampling-based methods have been developed to reduce the number of elements requiring a privacy budget, with adaptive budget allocation based on element importance. Wang et al. [80] introduced RescueDP, a scheme for real-time publishing of spatio-temporal crowd-sourced data with w-event level CDP, integrating adaptive sampling, privacy budget allocation, dynamic grouping, perturbation, and filtering techniques. The adaptive sampling component adjusts sampling rates based on data changes, ensuring efficient resource utilization. The privacy budget allocation mechanism dynamically distributes the privacy budget for sampling points across successive timestamps. Zhang et al. [81] proposed Re-DPoctor, a real-time health data releasing scheme ensuring w-event level CDP, enhancing utility with a partition algorithm safeguarding health data patterns and improving privacy through adaptive sampling and budget allocation. He et al. [66] introduced a new privacy concept using condensed local differential privacy (CLDP) [67] for w-event level privacy, aiming to enhance utility. They save privacy budget from empty releases and reallocate it to released elements, and utilize a PID controller-based method for adaptive budget allocation. However, this approach may inadvertently disclose private information through omitted empty points.

5.1.2 Optimization Strategies

Leveraging the correlations present in time series, the pre-processing or post-processing methods can be applied to improve the utility of the perturbed data. Ren et al. [82] addressed privacy challenges in high-dimensional crowdsourced data by proposing LoPub, an LDP data publication algorithm under event-level privacy. They use expectation maximization (EM) and Lasso regression to efficiently estimate multivariate joint distributions, identifying attribute correlations to reduce data dimensionality for distribution learning speed and utility improvements. Wang et al. [83] proposed the methods, LoCop and DR_LoCop, for releasing high-dimensional crowdsourced data under event-level LDP. The methods comprise four integrated components: a transformation component that ensures LDP by hashing and randomizing the data, an estimation component that infers probability distributions from the resulting Bloom filter strings, a computation component that derives marginal distributions and captures dependencies among dimensions, and a sampling component that generates a new synthetic dataset based on the computed distributions and dependencies. Fioretto and Hentenryck [84] introduced OptStream, a method for releasing time series under w-event level, which extends to handling hierarchical streams like energy profiles, making it applicable beyond its original energy domain. OptStream involves four key steps: sampling points for private measurement, perturbing them for privacy, reconstructing non-sampled points, and post-processing with convex optimization to improve accuracy by redistributing added noise. Zhang et al. [85] introduced a method for releasing differentially private sequential data using first-order autoregressive processes under user-level privacy. Their approach estimates unreleased data from previously released data by leveraging learned correlations, without requiring prior knowledge. The estimated data is combined with the observed data and perturbed with calibrated noise at each timestamp, facilitating real-time data release. Li et al. [86] proposed a framework for locally private stream data release that employs shuffling and subsampling techniques. Their approach maintains utility in the context of continual data collection by sampling a subset of users at each timestamp. An optimal sample size is determined to reduce redundant data and enhance utility, and the framework incorporates pre-processing within the shuffler to mitigate bias arising from distributed sampling. Besides pre- or post-processing methods, Bao et al. [87] proposed a mechanism based on the assumption of data fluctuation. Since time series may not change significantly over time, this mechanism formalizes the correlation between elements, allowing a later element to be represented by previous elements. To protect privacy, noise is added to the correlation.

5.2 Synthesis Based Methods

Directly releasing users' data poses significant risks of privacy breaches. An alternative approach is to train a synthesis model under strict privacy conditions and then release the data generated by this model for downstream tasks. To synthesize time series data under DP, one method involves first estimating the relevant statistics and then generating the synthetic data based on these estimations. Additionally, the advent of Generative Adversarial Networks (GANs) under DP [88, 89] has facilitated the use of deep learning algorithms to generate data, thereby enhancing both privacy protection and data accuracy.

5.2.1 Synthesis Based on Statistics

Synthesis mechanisms based on statistical features first capture the statistical characteristics from datasets under DP. Subsequently, new data is generated according to these estimated statistics. He et al. [90] introduced an efficient polynomial-time algorithm for generating online differentially private synthetic data under event-level privacy from a continuous time series within the hypercube $[0, 1]^d$. The algorithm achieves near-optimal accuracy bounds in 1-Wasserstein distance and extends previous work to include Lipschitz queries. By utilizing an online hierarchical partitioning approach and a novel Inhomogeneous Sparse Counting Algorithm, the method maintains strong privacy guarantees while ensuring high utility for infinite time horizons. To achieve a higher privacy level, Bun et al. [91] focused on generating differentially private synthetic data through statistical estimation under user-level CDP. They proposed algorithms that maintain the accuracy of fixed time window and cumulative time queries, ensuring minimal error while preserving privacy. Their approach involves a two-stage process that combines noisy estimates with post-processing techniques to ensure consistency and accuracy in synthetic data generation.

5.2.2 Synthesis Based on Generative Models

In addition to statistics-based mechanisms, another method for synthesizing time series is through Generative Adversarial Networks (GANs). Unlike other data types, time series requires consideration of temporal correlations. Frigerio et al. [92] presented a framework for releasing high-quality open data while ensuring user privacy through DP, addressing both continuous and discrete data. By leveraging deep learning and generative models with long short-term memory networks, the framework maintains data utility and correlations, introducing innovations such as clipping decay to optimize performance. Wang et al. [93] introduced PART-GAN, a privacy-preserving generative model designed for time series augmentation and sharing. PART-GAN combines Conditional and Temporal Generative Adversarial Networks (CT-GAN) with differential privacy mechanisms, enabling the generation of unlimited synthetic data that addresses issues of incomplete and irregularly sampled time series. Torfi et al. [94] proposed a mechanism to generate high-quality synthetic health record data while ensuring privacy using Rényi Differential Privacy (RDP). Their framework combines convolutional autoencoders and convolutional generative adversarial networks (CGAN) to effectively handle both discrete and continuous data, preserving temporal and feature correlations. For specific applications, Lamp et al. [95] introduced GlucoSynth, a novel privacy-preserving GAN framework designed to generate high-quality synthetic glucose traces while maintaining strong differential privacy guarantees. By focusing on preserving the relationships among glucose events (motifs) and temporal dynamics, GlucoSynth addresses the unique challenges of synthesizing glucose data.

5.3 Discussion

The aforementioned release mechanisms modify the values of the original time series, which can degrade utility in value-critical scenarios. For elements in a time series where occurrence indicates more sensitive information, temporal perturbation can be employed to avoid distorting the original values. Ye et al. [74] first proposed a method to achieve temporal perturbation in the local setting, maintaining the privacy guarantee while enhancing utility. As illustrated in Fig. 6, unlike value perturbation that directly modifies the original values by

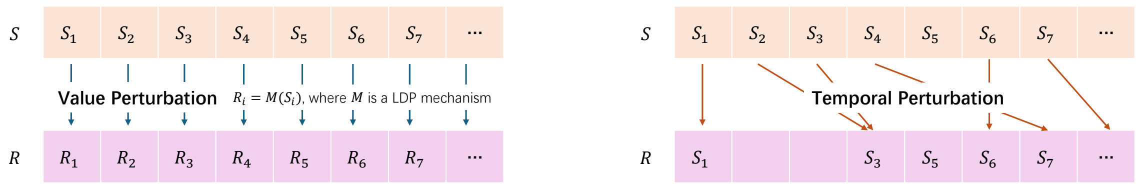

S
S_1 S_2 S_3 S_4 S_5 S_6 S_7 ⋯
Value Perturbation
$R_i = M(S_i)$, where M is a LDP mechanism
R
R_1 R_2 R_3 R_4 R_5 R_6 R_7 ⋯
Temporal Perturbation
S_1 S_3 S_5 S_6 S_7 ⋯

Figure 6: Value perturbation-based methods will perturb the original values by adding DP noise. In contrast, temporal perturbation-based methods will dispatch the values to corresponding timestamps for release.

adding LDP noise, temporal perturbation dispatches elements across different timestamps. This obfuscates the precise occurrence times, preventing adversaries from determining the exact event timings. However, temporal perturbation can lead to issues such as delayed releases, empty releases (where no elements are dispatched to certain timestamps), and element substitutions, resulting in missing data. To address these issues, Ye et al. [75] proposed a bi-directional perturbation mechanism that eliminates collisions during the dispatching process, ensuring that elements are only delayed. Furthermore, Mao et al. [96] extended the definition to a metric-based version tailored for anomaly detection, aiming to reduce collisions involving anomalous elements. However, these proposed mechanisms [74, 75, 96] primarily address event-level privacy concerns, leaving room for enhancements to achieve higher levels of privacy.

6 Differential Privacy for Location Based Services

Location Based Service (LBS) is a crucial application of mobile computing that provides personalized services based on users' geographical locations. These services, ranging from navigation assistance to location-based recommendations, require continuous collection and analysis of users' location data, often in the form of trajectories.

A trajectory is a specific type of time series that comprises spatial-temporal data. It can be regarded as a sequence of time-ordered points, denoted as $T = \{p_1, p_2, \cdots, p_T\}$, where p_i represents a location and T is the length of the trajectory. Compared with other time series, the correlations within trajectory data are more pronounced due to the constraints imposed by spatial variation. Regarding privacy levels for trajectory data, location privacy corresponds to event-level privacy, providing protection for individual locations, whereas trajectory privacy safeguards the entire trajectory.

In this section, we introduce mechanisms for handling trajectory data under differential privacy, organized according to utility improvement in location perturbation, privacy preservation against temporal correlation, and trajectory release. These mechanisms ensure that the privacy of individual locations and movements is preserved while maintaining the utility of the data for analysis and service provision.

6.1 Location Perturbation Based on Geo-indistinguishability

For meaningful outputs in LBS, the perturbed location should not deviate excessively from the actual one. As illustrated in Fig. 7, constraining the perturbation domain is essential for improving utility; otherwise, a large perturbation domain yields less useful results. For example, perturbing Paris to London is impractical [29]. Therefore, a metric-based privacy notion, ϵ-geo-indistinguishability, is proposed. Specifically, a user's level of privacy is defined as $\ell = \epsilon r$, where r is the radius of the perturbation domain, corresponding to r_i in Fig. 7. Here is the formal definition of geo-indistinguishability.

Definition 6.1: *[Geo-indistinguishability [29]] Given any two locations x and x' ($\mathrm{d}(x, x') \leq r$), a randomized mechanism M satisfies ϵ-geo-indistinguishability iff*

$$\Pr[M(x) \in Z] \leq e^{\epsilon d(x,x')} \Pr[M(x') \in Z],$$

where $Z \subseteq \mathcal{Z}$ is the possible output domain, where $\mathrm{d}(x, x')$ represents a distance measure between x and x'.

Since the inception of geo-indistinguishability, numerous enhancements have been made to improve the privacy notion from various perspectives. To enhance the calculation efficiency, Bordenabe et al. [97] proposed a method to optimize the trade-off between geo-indistinguishability and service quality in location privacy, employing linear programming to minimize service quality loss while ensuring optimal privacy guarantees. By reducing the number of constraints from cubic to quadratic, their approach significantly improves computational efficiency. Building on the concept of geo-indistinguishability, Weggenmann and Kerschbaum [98] introduced the notion of directional privacy, a relaxation of pure differential privacy that performs effectively in the local model. To enhance practical utility, Zhao et al. [99] proposed geo-ellipse-indistinguishability to protect individual location data in directional distribution analysis. This method incorporates the covariance matrix to account for dispersion and orientation of community locations, using elliptical noise instead of circular noise. The proposed mechanisms, based on gamma and multivariate normal distributions, ensure higher probabilities of randomized locations aligning with community location trends while maintaining statistical quality. Liang and Yi [100] further advanced the concept of geo-indistinguishability by introducing concentrated geo-privacy, an update from the CDP version. This approach supports advanced composition mechanisms for high-dimensional data and achieves a lower noise scale, thereby enhancing overall privacy protection while maintaining utility. Zhao and Chen [101] proposed vector-indistinguishability (vector-ind) to enhance location privacy by preserving distance

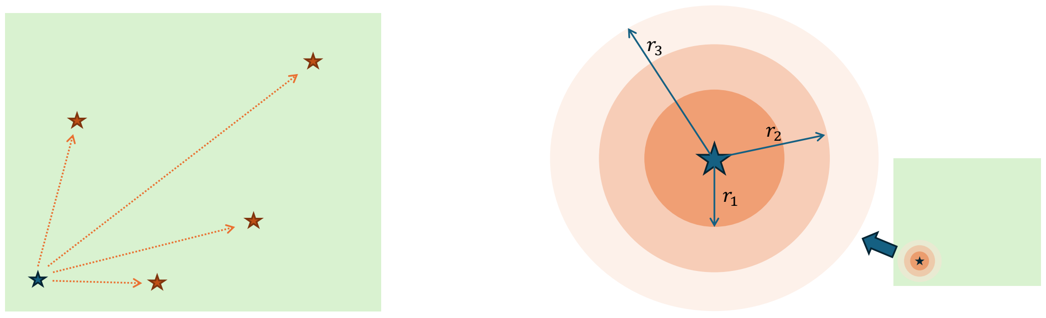


Figure 7: A traditional DP mechanism (illustrated in the left-hand figure) involves a perturbation domain (green area) for a location that is typically large, resulting in low utility for the perturbed location. To improve utility while providing useful services, a metric-based privacy notion, geo-indistinguishability, is introduced to control the perturbation (illustrated in the right-hand figure). A smaller distance r_i leads to higher utility but offers less privacy. Therefore, it is important to balance the trade off between utility and privacy when designing mechanisms under geo-indistinguishability.

and direction dependencies between successive locations. They introduced four mechanisms using Laplace and Uniform distributions to achieve vector-ind, maintaining data utility while ensuring CDP.

Several mechanisms have been proposed to address various privacy issues across numerous scenarios based on geo-indistinguishability. Yu et al. [102] proposed PIVE, a dynamic differential location privacy framework that integrates geo-indistinguishability and expected inference error to protect against inference attacks. PIVE operates in two phases: First, it identifies a protection location set based on user-defined error thresholds and prior knowledge; And then, it generates pseudo-locations within this set, ensuring differential privacy. This approach enables adaptive, personalized privacy settings tailored to individual user needs and location-based service requirements, thereby enhancing both privacy preservation and utility. Cao et al. [103] extended differential privacy to define ϵ-spatiotemporal event privacy and proposed a framework to quantify its protection level in existing location privacy-preserving mechanisms. They demonstrated their framework by adapting the Planar Laplace Mechanism for geo-indistinguishability to ensure spatiotemporal event privacy while maintaining linear computational complexity. Niu et al. [104] introduced Eclipse, a mechanism that combines geo-indistinguishability, k-anonymity, and expected inference error to protect location privacy against long-term observation attacks. Eclipse obfuscates user locations within an anonymity set, minimizing privacy leakage while maintaining service usability and correctness. Qiu et al. [105] tackled the Vehicle-based spatial crowdsourcing Location Privacy (VLP) problem, aiming to minimize travel cost distortion while preserving location privacy over road networks. They redefined geo-indistinguishability based on path distance and approximated the VLP problem as a linear programming formulation through discretization. To improve time efficiency, they proposed a two-layer optimization algorithm and analyzed the trade-off between privacy and quality of service. Haydari et al. [106] proposed a differential privacy-based map-matching algorithm (DPMM) for protecting user privacy in mobility data. DPMM generates privatized link-level location trajectories by incorporating road characteristics such as capacity and functional role. The algorithm adaptively selects the noise level based on link density, effectively balancing privacy preservation and trajectory accuracy.

6.2 Privacy Preservation Against Temporal Correlation

Due to the intrinsic features of location data, temporal correlations pose significant privacy issues when handling locations. Specifically, an adversary can infer information about a location based on its preceding or succeeding elements. For instance, Shao et al. [20] proposed iTracker, a framework designed to recover multiple trajectories from differentially private data using a structured sparsity model. iTracker leverages interdependencies among locations to enhance recovery accuracy, effectively challenging existing Laplace perturbation-based location protection mechanisms. To address privacy risks posed by temporal correlations in location data, numerous mechanisms have been proposed. Xiao and Xiong [107] introduced a solution that preserves location privacy with rigorous differential privacy guarantees by proposing δ-location set, which accounts for temporal correlations in location data. They also introduced the sensitivity hull to capture geometric sensitivity in multidimensional space and presented the planar isotropic mechanism (PIM), an efficient location perturbation method that achieves optimal utility while meeting differential privacy requirements. Cao et al. [21] first investigated the privacy loss of CDP mechanisms under temporal correlations and introduced the concept of Temporal Privacy Leakage (TPL). They proposed an efficient algorithm to calculate TPL and designed methods to convert traditional DP mechanisms to ones that mitigate TPL, ensuring privacy over continuous data releases by bounding the leakage within a defined parameter α. Xiao et al. [22] proposed LocLok, a system that protects user locations with differential privacy by modeling temporal correlations using a hidden Markov model and applying PIM for optimal noise addition. LocLok generates possible locations via the Markov model, perturbs them with PIM, and infers true locations within a set of all possible locations, ensuring robust local privacy even when adversaries have access to historical location data. Liu et al. [108] protected location privacy by analyzing the impact of temporal-spatial correlations and proposing new privacy definitions, introducing Bayesian-based geo-indistinguishability to better evaluate and enhance privacy levels. Their method optimally allocates noise among spatially and temporally correlated locations, effectively protecting sensitive locations within a trajectory while achieving differential privacy. Ma et al. [109] proposed RPTR under w-event level CDP to protect real-time vehicle trajectory data. They employed dynamic sampling and ensemble Kalman filters, utilizing a position transfer probability matrix to infer correlations and ensure accurate predictions while balancing data availability and privacy. Additionally, they introduced a regional privacy weight mechanism to enhance protection in high-density areas, thereby ensuring higher prediction accuracy and adaptability across different scenarios. Cao et al. [110] propose a post-processing framework to enhance the utility of differentially private streaming data releases by leveraging temporal correlations. They modeled the problem as a maximum a posterior estimation, transformed it into a nonlinear constrained programming problem, and used a transition matrix to incorporate both probabilistic and deterministic constraints. Ahuja et al. [111] proposed a method to release histogram information from trajectories under user-level CDP. To enhance utility, they introduced a method based on variational autoencoders to refine the histograms by utilizing the correlations of histograms.

6.3 Trajectory Release Based on Perturbation or Synthesis

As location-based services (LBS) become increasingly integral to everyday applications, ensuring the privacy of users while maintaining the utility of location data remains a critical challenge. Various mechanisms have been proposed to address this issue, each focusing on different aspects of location privacy and data utility. Wang et al. [112] introduced L-SRR, the first LDP framework for location-based services, enhancing utility while ensuring strict privacy. The proposed staircase randomized response mechanism perturbs user locations using optimized probabilities, significantly improving utility for applications such as traffic density estimation and k-nearest neighbor queries. Cunningham [113] introduced a locally differentially private mechanism for trajectory data sharing that integrates public knowledge to enhance utility while ensuring privacy. This mechanism perturbs hierarchically-structured n-grams of trajectory data to capture spatio-temporal relationships, leveraging public data without compromising privacy. Zhang et al. [114] proposed a trajectory perturbation mechanism under user-level LDP that enhances privacy by using adjacent direction information to connect neighboring points. They introduce a two-stage pivot sampling process utilizing bi-directional clues from pivots, and an anchor-based method to restrict the spatial region of trajectories.

Synthetic trajectory generation has emerged as another promising solution, allowing for the publication of useful data without compromising individual privacy. Gursoy et al. [115] presented DP-Star, a framework for publishing trajectory data that ensures differential privacy while maintaining high utility. DP-Star normalizes raw trajectories using representative points, constructs a density-aware grid to preserve spatial densities, and employs a private Markov mobility model to maintain correlations and intra-trajectory mobility. This results in synthetic trajectory datasets that are both privacy-preserving and useful for various data mining tasks. Moreover, Gursoy et al. [116] presented AdaTrace, a scalable location trace synthesizer that achieves statistical privacy, deterministic attack resilience, and strong utility preservation. AdaTrace generates differentially private synthetic traces through a four-phase process: feature extraction, noise injection, and utility-aware synthesis. The synthetic traces preserve utility-critical information and are robust against Bayesian inference, partial sniffing, and outlier leakage attacks, ensuring privacy without significant utility loss. Du et al. [117] introduced LDPTrace, a locally differentially private framework for synthesizing realistic trajectories with minimal computational cost and strong privacy guarantees. LDPTrace captures key movement patterns from users' trajectories, ensuring robust statistical privacy and resilience against attacks. Extensive evaluations demonstrate that LDPTrace generates authentic trajectories without external knowledge, outperforming existing methods in terms of utility and privacy protection. Hu et al. [118] introduced RetraSyn under w-event level LDP, aimed at real-time trajectory synthesis while ensuring data privacy. RetraSyn leverages mobility patterns from trajectory streams and incorporates a global mobility model, dynamic update mechanisms, and Markov-based synthesis to generate realistic trajectories. This framework effectively captures complex spatial-temporal contexts and employs adaptive privacy budget allocation strategies, ensuring authenticity and practicality in diverse real-world scenarios. Sun et al. [119] proposed SPRT, a method for synthesizing private and realistic vehicle trajectories by incorporating geographic structures into differential privacy mechanisms. SPRT constructs a geography-aware grid to capture accurate mobility patterns and defines a moveable constraint based on real-world conditions, enhancing both summary-level statistics and individual-level mobility patterns.

7 Open Challenges

Although many mechanisms have been proposed to handle time series under differential privacy, several issues still need to be addressed. In this section, the challenges will be introduced according to privacy model, potential attacks, data type, and learning based problems.

7.1 Privacy Model

The privacy model is a crucial factor in differential privacy. As aforementioned, there are three privacy levels when handling time series [19]. Event-level privacy guarantees the privacy of a single element in a time series, making it easier to implement since the sensitivity of an individual element in neighboring datasets is simpler to measure. In contrast, user-level privacy provides a higher privacy guarantee and is more practical in real-world applications. However, bounding the sensitivity of a single user's participation is challenging, making the allocation of the privacy budget more complex. Additionally, utility issues become more pronounced when dealing with infinite time series.

Several research works have explored handling infinite time series under user-level privacy with specific conditions. Dong et al. [48] introduced mechanisms for basic queries such as count and sum. These mechanisms are based on event-level privacy approaches and are adapted to user-level privacy by bounding the maximum changes of a single user in a time series. For LDP, Xue et al. [64] proposed a mechanism for count queries, but it requires that the time series does not fluctuate significantly. Feng et al. [68] proposed a strategy that randomly allocates the privacy budget according to a converging sum series.

Therefore, improving the utility of infinite time series under user-level differential privacy is an intriguing future direction. Beyond basic queries, efforts can be made to accommodate specific queries or applications. Key challenges include accurately measuring data sensitivity and effectively allocating the privacy budget. Overcoming these challenges can enhance the practical utility of differential privacy mechanisms for managing infinite time series under user-level privacy.

7.2 Temporal Correlation Based Attacks

Compared to other data types, the correlation in time series is more pronounced. Due to the inherent sequential nature of time series, each element is often directly influenced by its predecessors. Various works have employed the Markov model to capture and represent these correlations under DP [21, 22, 115]. In the context of the Markov model, an element is directly influenced only by its immediate neighboring element. Consequently, the influence of previous elements is implicitly carried forward through the chain of direct dependencies between neighboring elements. Since time series are always modeled explicitly, this simplicity can result in the model overlooking long-term information that extend beyond immediate neighbors. To capture such information, more complex models like the long short-term memory network [3] are needed. Moreover, for specific types of time series such as trajectories, public knowledge can introduce additional privacy issues. For example, certain perturbations may be impossible due to physical world limitations. In summary, compared with single data points, elements in time series are at a greater risk of privacy leakage. This suggests a potential direction for research: attacking existing privacy mechanisms by exploiting these correlations and to design new mechanisms that account for the inherent dependencies in time series.

7.3 Complex Data Type

Most current works can only handle simple time series with high utility, such as one value at each timestamp. However, real-world data are more complex, and mechanisms should be designed to handle this complexity. Here are two examples:

First, sensor data are often multi-dimensional and correlated across each dimension. This complexity requires mechanisms capable of managing and analyzing data with multiple interacting variables. Traditional methods that handle single-dimensional elements at each timestamp are insufficient for capturing the nuances of multi-dimensional sensor data. For example, environmental sensors may collect temperature, humidity, and air pressure simultaneously. Analyzing these factors independently can miss critical interactions and patterns, such as how temperature changes might influence humidity levels during users' activities. Therefore, advanced methods must be developed to process and interpret multi-dimensional sensor data effectively.

Second, the element at each timestamp can be intricate. For instance, social networks change over time, making real-time analysis of such dynamic networks complicated. Managing the evolution of relationships and interactions within the network adds a layer of complexity beyond time series analysis. Additionally, privacy concerns in such contexts extend beyond the temporal dimension to include graph privacy, encompassing node-level privacy and edge-level privacy. Mechanisms must account for these additional privacy requirements to ensure data protection while enabling real-time analysis.

7.4 Learning Based Problems

Current DP mechanisms are primarily designed for basic queries, such as count, mean, and frequency. However, time series without privacy concerns are often used for more complex downstream tasks, such as classification and clustering. These tasks require a deeper understanding and manipulation of the data, going beyond simple statistical queries. To accommodate more practical downstream tasks, it is essential to develop DP mechanisms that can support these sophisticated operations effectively, ensuring both the utility and privacy of the data.

A reasonable solution is to generate synthetic time series from the real dataset. Synthetic approach allows the fundamental features of the time series to be captured while protecting the privacy of the underlying data. Since the features of time series are complex and extend beyond basic statistics, traditional statistical methods are insufficient for capturing these intricate patterns. For example, critical patterns such as seasonal trends, cyclic behaviors, and sudden anomalies are vital in time series analysis but are not adequately addressed by basic statistical methods.

Therefore, learning-based methods are preferable for generating synthetic time series. These methods can model and replicate the complex dependencies and structures inherent in time series. For instance, the method proposed by Lamp et al. [95] for synthesizing glucose traces exemplifies how deep learning can be applied to generate realistic and privacy-preserving synthetic data. Predictably, more and more application-specific mechanisms will be proposed.

8 Conclusion

In this paper, we present a comprehensive survey on handling time series under differential privacy. We begin by introducing the basic concepts of time series and differential privacy, along with relevant definitions. Our survey starts with an exploration of two basic queries: count queries and sum/mean queries. For each query type, we first explain the concept of the basic query, then review the core techniques or developments related to the queries, and finally discuss the advanced queries derived from the basic ones. At the end of each query section, we review the downstream tasks based on these queries. Subsequently, we introduce mechanisms for time series release, categorizing them into value perturbation based methods and synthetic generation based methods. Additionally, we dedicate a separate section to location-based services (LBS), as they are common application scenarios for time series. We review relevant papers for LBS according to two popular privacy issues and the demands of trajectory release. Finally, we illustrate four open challenges and suggest future directions.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant No: 62372122, 62072390, and 92270123), the Research Grants Council, Hong Kong SAR (Grant No: 15203120, 15208923 and 15210023), Shenzhen Key Laboratory of Safety and Security for Next Generation of Industrial Internet
(Grant No. ZDSYS20210623092007023) and Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation (Grant No. 2020B121201001)

References

[1] Hyndman, R. & Athanasopoulos, G. Forecasting: principles and practice. (OTexts,2018)

[2] Gardner Jr, E. Exponential smoothing: The state of the art—Part II. International Journal Of Forecasting. **22**, 637-666 (2006)

[3] Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W. & Woo, W. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. Advances In Neural Information Processing Systems. **28** (2015)

[4] C. Dwork, "Differential privacy," in International colloquium on automata, languages, and programming. Springer, 2006, pp. 1–12.

[5] Ye, Q. & Hu, H. Local differential privacy: Tools, challenges, and opportunities. International Conference On Web Information Systems Engineering. pp. 13-23 (2020)

[6] M. Yang, T. Guo, T. Zhu, I. Tjuawinata, J. Zhao, and K.-Y. Lam, "Local differential privacy and its applications: A comprehensive survey," Computer Standards & Interfaces, p. 103827, 2023.

[7] Wang, T., Blocki, J., Li, N. & Jha, S. Locally differentially private protocols for frequency estimation. 26th USENIX Security Symposium (USENIX Security 17). pp. 729-745 (2017)

[8] Wang, Z., Pang, X., Chen, Y., Shao, H., Wang, Q., Wu, L., Chen, H. & Qi, H. Privacy-preserving crowd-sourced statistical data publishing with an untrusted server. IEEE Transactions On Mobile Computing. **18**, 1356-1367 (2018)

[9] Wang, T., Li, N. & Jha, S. Locally differentially private heavy hitter identification. IEEE Transactions On Dependable And Secure Computing. **18**, 982-993 (2019)

[10] Fu, Y., Ye, Q., Du, R. & Hu, H. Collecting Multi-type and Correlation-Constrained Streaming Sensor Data with Local Differential Privacy. ACM Transactions On Sensor Networks. (2023)

[11] Li, Z., Wang, T., Lopuhaä-Zwakenberg, M., Li, N. & Škoric, B. Estimating numerical distributions under local differential privacy. Proceedings Of The 2020 ACM SIGMOD International Conference On Management Of Data. pp. 621-635 (2020)

[12] Ye, Q., Hu, H., Au, M., Meng, X. & Xiao, X. Towards locally differentially private generic graph metric estimation. 2020 IEEE 36th International Conference On Data Engineering (ICDE). pp. 1922-1925 (2020)

[13] Zhang, J., Zhang, Z., Xiao, X., Yang, Y. & Winslett, M. Functional mechanism: Regression analysis under differential privacy. ArXiv Preprint ArXiv:1208.0219. (2012)

[14] Abadi, M., Chu, A., Goodfellow, I., McMahan, H., Mironov, I., Talwar, K. & Zhang, L. Deep learning with differential privacy. Proceedings Of The 2016 ACM SIGSAC Conference On Computer And Communications Security. pp. 308-318 (2016)

[15] Fu, J., Ye, Q., Hu, H., Chen, Z., Wang, L., Wang, K. & Xun, R. DPSUR: Accelerating Differentially Private Stochastic Gradient Descent Using Selective Update and Release. ArXiv Preprint ArXiv:2311.14056. (2023)

[16] Dwork, C., Naor, M., Pitassi, T. & Rothblum, G. Differential privacy under continual observation. Proceedings Of The Forty-second ACM Symposium On Theory Of Computing. pp. 715-724 (2010)

[17] Chan, T., Shi, E. & Song, D. Private and continual release of statistics. ACM Transactions On Information And System Security (TISSEC). **14**, 1-24 (2011)

[18] Fan, L. & Xiong, L. An adaptive approach to real-time aggregate monitoring with differential privacy. IEEE Transactions On Knowledge And Data Engineering. **26**, 2094-2106 (2013)

[19] Kellaris, G., Papadopoulos, S., Xiao, X. & Papadias, D. Differentially private event sequences over infinite streams. Proc. VLDB Endow.. **7**, 1155-1166 (2014,8), https://doi.org/10.14778/2732977.2732989

[20] Shao, M., Li, J., Yan, Q., Chen, F., Huang, H. & Chen, X. Structured sparsity model based trajectory tracking using private location data release. IEEE Transactions On Dependable And Secure Computing. **18**, 2983-2995 (2020)

[21] Cao, Y., Yoshikawa, M., Xiao, Y. & Xiong, L. Quantifying differential privacy under temporal correlations. 2017 IEEE 33rd International Conference On Data Engineering (ICDE). pp. 821-832 (2017)

[22] Xiao, Y., Xiong, L., Zhang, S. & Cao, Y. Loclok: Location cloaking with differential privacy via hidden markov model. Proceedings Of The VLDB Endowment. **10**, 1901-1904 (2017)

[23] Dwork, C., Roth, A. & Others The algorithmic foundations of differential privacy. Foundations And Trends® In Theoretical Computer Science. **9**, 211-407 (2014)

[24] Zhao, Y. & Chen, J. A survey on differential privacy for unstructured data content. ACM Computing Surveys (CSUR). **54**, 1-28 (2022)

[25] Zhao, Y., Du, J. & Chen, J. Scenario-based Adaptations of Differential Privacy: A Technical Survey. ACM Computing Surveys. **56**, 1-39 (2024)

[26] Miranda-Pascual, À., Guerra-Balboa, P., Parra-Arnau, J., Forné, J. & Strufe, T. SoK: Differentially private publication of trajectory data. Proceedings On Privacy Enhancing Technologies. (2023)

[27] Katsomallos, M., Tzompanaki, K. & Kotzinos, D. Privacy, space and time: A survey on privacy-preserving continuous data publishing. Journal Of Spatial Information Science. **2019**, 57-103 (2019)

[28] Li, C., Miklau, G., Hay, M., McGregor, A. & Rastogi, V. The matrix mechanism: optimizing linear counting queries under differential privacy. The VLDB Journal. **24** pp. 757-781 (2015)

[29] Andrés, M., Bordenabe, N., Chatzikokolakis, K. & Palamidessi, C. Geo-indistinguishability: Differential privacy for location-based systems. Proceedings Of The 2013 ACM SIGSAC Conference On Computer & Communications Security. pp. 901-914 (2013)

[30] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," Advances in Neural Information Processing Systems, vol. 30, 2017.

[31] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in Proceedings of the 2014 ACM SIGSAC conference on computer and communications security, 2014, pp. 1054–1067.

[32] A. G. Thakurta, A. H. Vyrros, U. S. Vaishampayan, G. Kapoor, J. Freudiger, V. R. Sridhar, and D. Davidson, "Learning new words," Mar. 14 2017, uS Patent 9,594,741.

[33] C. Dwork, "Differential Privacy and the US Census," in Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI symposium on principles of database systems, 2019, pp. 1–1.

[34] Li, N., Lyu, M., Su, D. & Yang, W. Differential privacy: From theory to practice. (Springer,2017)

[35] Kifer, D. & Machanavajjhala, A. Pufferfish: A framework for mathematical privacy definitions. ACM Transactions On Database Systems (TODS). **39**, 1-36 (2014)

[36] Song, S., Wang, Y. & Chaudhuri, K. Pufferfish privacy mechanisms for correlated data. Proceedings Of The 2017 ACM International Conference On Management Of Data. pp. 1291-1306 (2017)

[37] Korolova, A., Kenthapadi, K., Mishra, N. & Ntoulas, A. Releasing search queries and clicks privately. Proceedings Of The 18th International Conference On World Wide Web. pp. 171-180 (2009)

[38] Xiao, X., Wang, G. & Gehrke, J. Differential privacy via wavelet transforms. IEEE Transactions On Knowledge And Data Engineering. **23**, 1200-1214 (2010)

[39] Wang, N., Xiao, X., Yang, Y., Zhao, J., Hui, S., Shin, H., Shin, J. & Yu, G. Collecting and analyzing multidimensional data with local differential privacy. 2019 IEEE 35th International Conference On Data Engineering (ICDE). pp. 638-649 (2019)

[40] Xue, Q., Zhu, Y. & Wang, J. Mean estimation over numeric data with personalized local differential privacy. Frontiers Of Computer Science. **16** pp. 1-10 (2022)

[41] Zhou, M., Wang, T., Chan, T., Fanti, G. & Shi, E. Locally differentially private sparse vector aggregation. 2022 IEEE Symposium On Security And Privacy (SP). pp. 422-439 (2022)

[42] Wei, F., Bao, E., Xiao, X., Yang, Y. & Ding, B. AAA: an Adaptive Mechanism for Locally Differential Private Mean Estimation. ArXiv Preprint ArXiv:2404.01625. (2024)

[43] Ye, Q., Hu, H., Au, M., Meng, X. & Xiao, X. LF-GDPR: A framework for estimating graph metrics with local differential privacy. IEEE Transactions On Knowledge And Data Engineering. **34**, 4905-4920 (2020)

[44] Ma, Y., Zhang, H., Cai, Y. & Yang, H. Decision tree for locally private estimation with public data. Advances In Neural Information Processing Systems. **36** (2024)

[45] Esling, P. & Agon, C. Time-series data mining. ACM Computing Surveys (CSUR). **45**, 1-34 (2012)

[46] Silva, J., Faria, E., Barros, R., Hruschka, E., Carvalho, A. & Gama, J. Data stream clustering: A survey. ACM Computing Surveys (CSUR). **46**, 1-31 (2013)

[47] Ruiz, A., Flynn, M., Large, J., Middlehurst, M. & Bagnall, A. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining And Knowledge Discovery. **35**, 401-449 (2021)

[48] Dong, W., Luo, Q. & Yi, K. Continual Observation under User-level Differential Privacy. 2023 IEEE Symposium On Security And Privacy (SP). pp. 2190-2207 (2023)

[49] Henzinger, M., Upadhyay, J. & Upadhyay, S. Almost tight error bounds on differentially private continual counting. Proceedings Of The 2023 Annual ACM-SIAM Symposium On Discrete Algorithms (SODA). pp. 5003-5039 (2023)

[50] Cardoso, A. & Rogers, R. Differentially private histograms under continual observation: Streaming selection into the unknown. International Conference On Artificial Intelligence And Statistics. pp. 2397-2419 (2022)

[51] Knop, A. & Steinke, T. Counting Distinct Elements Under Person-Level Differential Privacy. 37th Conference On Neural Information Processing Systems (NeurIPS 2023). (2023)

[52] Bun, M. & Steinke, T. Concentrated differential privacy: Simplifications, extensions, and lower bounds. Theory Of Cryptography Conference. pp. 635-658 (2016)

[53] Kalemaj, I., Jain, P., Raskhodnikova, S., Sivakumar, S. & Smith, A. Counting Distinct Elements in the Turnstile Model with Differential Privacy under Continual Observation. Advances In Neural Information Processing Systems, NeurIPS 2023. (2023)

[54] Epasto, A., Mao, J., Medina, A., Mirrokni, V., Vassilvitskii, S. & Zhong, P. Differentially private continual releases of streaming frequency moment estimations. ArXiv Preprint ArXiv:2301.05605. (2023)

[55] Zhang, B., Doroshenko, V., Kairouz, P., Steinke, T., Thakurta, A., Ma, Z., Apte, H. & Spacek, J. Differentially Private Stream Processing at Scale. ArXiv Preprint ArXiv:2303.18086. (2023)

[56] Koga, T., Meehan, C. & Chaudhuri, K. Privacy amplification by subsampling in time domain. International Conference On Artificial Intelligence And Statistics. pp. 4055-4069 (2022)

[57] Fan, L. & Xiong, L. Differentially private anomaly detection with a case study on epidemic outbreak detection. 2013 IEEE 13th International Conference On Data Mining Workshops. pp. 833-840 (2013)

[58] Li, H., Xiong, L., Jiang, X. & Liu, J. Differentially private histogram publication for dynamic datasets: an adaptive sampling approach. Proceedings Of The 24th ACM International On Conference On Information And Knowledge Management. pp. 1001-1010 (2015)

[59] Wang, Q., Lu, X., Zhang, Y., Wang, Z., Qin, Z. & Ren, K. Secweb: Privacy-preserving web browsing monitoring

with w-event differential privacy. Security And Privacy In Communication Networks: 12th International Conference, SecureComm 2016, Guangzhou, China, October 10-12, 2016, Proceedings 12. pp. 454-474 (2017)

[60] Liang, W., Chen, H., Liu, R., Wu, Y. & Li, C. A pufferfish privacy mechanism for monitoring web browsing behavior under temporal correlations. Computers & Security. **92** pp. 101754 (2020)

[61] Ding, J., Ghosh, A., Sarkar, R. & Gao, J. Publishing Asynchronous Event Times with Pufferfish Privacy. 2022 18th International Conference On Distributed Computing In Sensor Systems (DCOSS). pp. 53-60 (2022)

[62] Erlingsson, Ú., Pihur, V. & Korolova, A. Rappor: Randomized aggregatable privacy-preserving ordinal response. Proceedings Of The 2014 ACM SIGSAC Conference On Computer And Communications Security. pp. 1054-1067 (2014)

[63] Ding, B., Kulkarni, J. & Yekhanin, S. Collecting telemetry data privately. Advances In Neural Information Processing Systems. **30** (2017)

[64] Xue, Q., Ye, Q., Hu, H., Zhu, Y. & Wang, J. DDRM: A continual frequency estimation mechanism with local differential privacy. IEEE Transactions On Knowledge And Data Engineering. (2022)

[65] Arcolezi, H., Pinzón, C., Palamidessi, C. & Gambs, S. Frequency estimation of evolving data under local differential privacy. ArXiv Preprint ArXiv:2210.00262. (2022)

[66] He, Y., Wang, F., Deng, X., Ni, J., Feng, J. & Liu, S. Ordinal data stream collection with condensed local differential privacy. 2022 IEEE 24th Int Conf On High Performance Computing & Communications; 8th Int Conf On Data Science & Systems; 20th Int Conf On Smart City; 8th Int Conf On Dependability In Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys). pp. 562-569 (2022)

[67] Gursoy, M., Tamersoy, A., Truex, S., Wei, W. & Liu, L. Secure and utility-aware data collection with condensed local differential privacy. IEEE Transactions On Dependable And Secure Computing. **18**, 2365-2378 (2019)

[68] Feng, S., Mohammady, M., Wang, H., Li, X., Qin, Z. & Hong, Y. DPI: Ensuring Strict Differential Privacy for Infinite Data Streaming. ArXiv Preprint ArXiv:2312.04738. (2023)

[69] Li, X., Liu, W., Lou, J., Hong, Y., Zhang, L., Qin, Z. & Ren, K. Local differentially private heavy hitter detection in data streams with bounded memory. Proceedings Of The ACM On Management Of Data. **2**, 1-27 (2024)

[70] Gu, H., Plagemann, T., Benndorf, M., Goebel, V. & Koldehofe, B. Differential Privacy for Protecting Private Patterns in Data Streams. 2023 IEEE 39th International Conference On Data Engineering Workshops (ICDEW). pp. 118-124 (2023)

[71] Bolot, J., Fawaz, N., Muthukrishnan, S., Nikolov, A. & Taft, N. Private decayed predicate sums on streams. Proceedings Of The 16th International Conference On Database Theory. pp. 284-295 (2013)

[72] Henzinger, M., Upadhyay, J. & Upadhyay, S. A unifying framework for differentially private sums under continual observation. Proceedings Of The 2024 Annual ACM-SIAM Symposium On Discrete Algorithms (SODA). pp. 995-1018 (2024)

[73] Wang, T., Chen, J., Zhang, Z., Su, D., Cheng, Y., Li, Z., Li, N. & Jha, S. Continuous release of data streams under both centralized and local differential privacy. Proceedings Of The 2021 ACM SIGSAC Conference On Computer And Communications Security. pp. 1237-1253 (2021)

[74] Ye, Q., Hu, H., Li, N., Meng, X., Zheng, H. & Yan, H. Beyond value perturbation: Local differential privacy in the temporal setting. IEEE INFOCOM 2021-IEEE Conference On Computer Communications. pp. 1-10 (2021)

[75] Ye, Q., Hu, H., Huang, K., Au, M. & Xue, Q. Stateful switch: Optimized time series release with local differential privacy. IEEE INFOCOM 2023-IEEE Conference On Computer Communications. pp. 1-10 (2023)

[76] Perrier, V., Asghar, H. & Kaafar, D. Private continual release of real-valued data streams. ArXiv Preprint ArXiv:1811.03197. (2018)

[77] Wang, Z., Liu, W., Pang, X., Ren, J., Liu, Z. & Chen, Y. Towards pattern-aware privacy-preserving real-time data collection. IEEE INFOCOM 2020-IEEE Conference On Computer Communications. pp. 109-118 (2020)

[78] Kurt, M., Yılmaz, Y., Wang, X. & Mosterman, P. Online privacy-preserving data-driven network anomaly detection. IEEE Journal On Selected Areas In Communications. **40**, 982-998 (2022)

[79] Ren, X., Shi, L., Yu, W., Yang, S., Zhao, C. & Xu, Z. LDP-IDS: Local differential privacy for infinite data streams. Proceedings Of The 2022 International Conference On Management Of Data. pp. 1064-1077 (2022)

[80] Wang, Q., Zhang, Y., Lu, X., Wang, Z., Qin, Z. & Ren, K. RescueDP: Real-time spatio-temporal crowd-sourced data publishing with differential privacy. IEEE INFOCOM 2016-The 35th Annual IEEE International Conference On Computer Communications. pp. 1-9 (2016)

[81] Zhang, J., Liang, X., Zhang, Z., He, S. & Shi, Z. Re-DPoctor: Real-time health data releasing with w-day differential

privacy. GLOBECOM 2017-2017 IEEE Global Communications Conference. pp. 1-6 (2017)

[82] Ren, X., Yu, C., Yu, W., Yang, S., Yang, X., McCann, J. & Philip, S. LoPub : high-dimensional crowdsourced data publication with local differential privacy. IEEE Transactions On Information Forensics And Security. **13**, 2151-2166 (2018)

[83] Wang, T., Yang, X., Ren, X., Yu, W. & Yang, S. Locally private high-dimensional crowdsourced data release based on copula functions. IEEE Transactions On Services Computing. **15**, 778-792 (2019)

[84] Fioretto, F. & Van Hentenryck, P. Optstream: Releasing time series privately. Journal Of Artificial Intelligence Research. **65** pp. 423-456 (2019)

[85] Zhang, X., Khalili, M. & Liu, M. Differentially private real-time release of sequential data. ACM Transactions On Privacy And Security. **26**, 1-29 (2022)

[86] Li, X., Cao, Y. & Yoshikawa, M. Locally Private Streaming Data Release with Shuffling and Subsampling. 2023 IEEE 39th International Conference On Data Engineering Workshops (ICDEW). pp. 125-131 (2023)

[87] Bao, E., Yang, Y., Xiao, X. & Ding, B. CGM: an enhanced mechanism for streaming data collection with local differential privacy. Proceedings Of The VLDB Endowment. **14**, 2258-2270 (2021)

[88] Xie, L., Lin, K., Wang, S., Wang, F. & Zhou, J. Differentially private generative adversarial network. ArXiv Preprint ArXiv:1802.06739. (2018)

[89] Jordon, J., Yoon, J. & Van Der Schaar, M. PATE-GAN: Generating synthetic data with differential privacy guarantees. International Conference On Learning Representations. (2018)

[90] He, Y., Vershynin, R. & Zhu, Y. Online Differentially Private Synthetic Data Generation. ArXiv Preprint ArXiv:2402.08012. (2024)

[91] Bun, M., Gaboardi, M., Neunhoeffer, M. & Zhang, W. Continual Release of Differentially Private Synthetic Data from Longitudinal Data Collections. Proceedings Of The ACM On Management Of Data. **2**, 1-26 (2024)

[92] Frigerio, L., Oliveira, A., Gomez, L. & Duverger, P. Differentially private generative adversarial networks for time series, continuous, and discrete open data. ICT Systems Security And Privacy Protection: 34th IFIP TC 11 International Conference, SEC 2019, Lisbon, Portugal, June 25-27, 2019, Proceedings 34. pp. 151-164 (2019)

[93] Wang, S., Rudolph, C., Nepal, S., Grobler, M. & Chen, S. PART-GAN: Privacy-preserving time-series sharing. Artificial Neural Networks And Machine Learning–ICANN 2020: 29th International Conference On Artificial Neural Networks, Bratislava, Slovakia, September 15–18, 2020, Proceedings, Part I 29. pp. 578-593 (2020)

[94] Torfi, A., Fox, E. & Reddy, C. Differentially private synthetic medical data generation using convolutional GANs. Information Sciences. **586** pp. 485-500 (2022)

[95] Lamp, J., Derdzinski, M., Hannemann, C., Linden, J., Feng, L., Wang, T. & Evans, D. GlucoSynth: Generating Differentially-Private Synthetic Glucose Traces. Advances In Neural Information Processing Systems. **36** (2024)

[96] Mao, Y., Ye, Q., Wang, Q. & Hu, H. Utility-Aware Time Series Data Release with Anomalies under TLDP. IEEE Transactions On Mobile Computing. (2023)

[97] Bordenabe, N., Chatzikokolakis, K. & Palamidessi, C. Optimal geo-indistinguishable mechanisms for location privacy. Proceedings Of The 2014 ACM SIGSAC Conference On Computer And Communications Security. pp. 251-262 (2014)

[98] Weggenmann, B. & Kerschbaum, F. Differential privacy for directional data. Proceedings Of The 2021 ACM SIGSAC Conference On Computer And Communications Security. pp. 1205-1222 (2021)

[99] Zhao, Y., Yuan, D., Du, J. & Chen, J. Geo-ellipse-indistinguishability: community-aware location privacy protection for directional distribution. IEEE Transactions On Knowledge And Data Engineering. (2022)

[100] Liang, Y. & Yi, K. Concentrated geo-privacy. Proceedings Of The 2023 ACM SIGSAC Conference On Computer And Communications Security. pp. 1934-1948 (2023)

[101] Zhao, Y. & Chen, J. Vector-indistinguishability: location dependency based privacy protection for successive location data. IEEE Transactions On Computers. (2023)

[102] Yu, L., Liu, L. & Pu, C. Dynamic Differential Location Privacy with Personalized Error Bounds.. NDSS. (2017)

[103] Cao, Y., Xiao, Y., Xiong, L. & Bai, L. PriSTE: from location privacy to spatiotemporal event privacy. 2019 IEEE 35th International Conference On Data Engineering (ICDE). pp. 1606-1609 (2019)

[104] Niu, B., Chen, Y., Wang, Z., Li, F., Wang, B. & Li, H. Eclipse: Preserving differential location privacy against long-term observation attacks. IEEE Transactions On Mobile Computing. **21**, 125-138 (2020)

[105] Qiu, C., Squicciarini, A., Pang, C., Wang, N. & Wu, B. Location privacy protection in vehicle-based spatial crowdsourcing via geo-indistinguishability. IEEE Transactions On Mobile Computing. **21**, 2436-2450 (2020)

[106] Haydari, A., Chuah, C., Zhang, M., Macfarlane, J. & Peisert, S. Differentially private map matching for mobility trajectories. Proceedings Of The 38th Annual Computer Security Applications Conference. pp. 293-303 (2022)

[107] Xiao, Y. & Xiong, L. Protecting locations with differential privacy under temporal correlations. Proceedings Of The 22nd ACM SIGSAC Conference On Computer And Communications Security. pp. 1298-1309 (2015)

[108] Liu, B., Zhu, T., Zhou, W., Wang, K., Zhou, H. & Ding, M. Protecting privacy-sensitive locations in trajectories with correlated positions. 2019 IEEE Global Communications Conference (GLOBECOM). pp. 1-6 (2019)

[109] Ma, Z., Zhang, T., Liu, X., Li, X. & Ren, K. Real-time privacy-preserving data release over vehicle trajectory. IEEE Transactions On Vehicular Technology. **68**, 8091-8102 (2019)

[110] Cao, X., Cao, Y., Pappachan, P., Nakamura, A. & Yoshikawa, M. Differentially Private Streaming Data Release Under Temporal Correlations via Post-processing. IFIP Annual Conference On Data And Applications Security And Privacy. pp. 184-200 (2023)

[111] Ahuja, R., Zeighami, S., Ghinita, G. & Shahabi, C. A Neural Approach to Spatio-Temporal Data Release with User-Level Differential Privacy. Proceedings Of The ACM On Management Of Data. **1**, 1-25 (2023)

[112] Wang, H., Hong, H., Xiong, L., Qin, Z. & Hong, Y. L-srr: Local differential privacy for location-based services with staircase randomized response. Proceedings Of The 2022 ACM SIGSAC Conference On Computer And Communications Security. pp. 2809-2823 (2022)

[113] Cunningham, T., Cormode, G., Ferhatosmanoglu, H. & Srivastava, D. Real-world trajectory sharing with local differential privacy. ArXiv Preprint ArXiv:2108.02084. (2021)

[114] Zhang, Y., Ye, Q., Chen, R., Hu, H. & Han, Q. Trajectory data collection with local differential privacy. ArXiv Preprint ArXiv:2307.09339. (2023)

[115] Gursoy, M., Liu, L., Truex, S. & Yu, L. Differentially private and utility preserving publication of trajectory data. IEEE Transactions On Mobile Computing. **18**, 2315-2329 (2018)

[116] Gursoy, M., Liu, L., Truex, S., Yu, L. & Wei, W. Utility-aware synthesis of differentially private and attack-resilient location traces. Proceedings Of The 2018 ACM SIGSAC Conference On Computer And Communications Security. pp. 196-211 (2018)

[117] Du, Y., Hu, Y., Zhang, Z., Fang, Z., Chen, L., Zheng, B. & Gao, Y. Ldptrace: Locally differentially private trajectory synthesis. Proceedings Of The VLDB Endowment. **16**, 1897-1909 (2023)

[118] Hu, Y., Du, Y., Zhang, Z., Fang, Z., Chen, L., Zheng, K. & Gao, Y. Real-Time Trajectory Synthesis with Local Differential Privacy. ArXiv Preprint ArXiv:2404.11450. (2024)

[119] Sun, X., Ye, Q., Hu, H., Wang, Y., Huang, K., Wo, T. & Xu, J. Synthesizing realistic trajectory data with differential privacy. IEEE Transactions On Intelligent Transportation Systems. (2023)

A Review of Adaptive Techniques and Data Management Issues in DP-SGD

Islam A. Monir, Muhamad I. Fauzan, Gabriel Ghinita

Abstract

Differentially-Private Stochastic Gradient Descent (DP-SGD) established itself as the most prominent technique for training neural networks with formal privacy guarantees. Almost a decade after its introduction, DP-SGD has become the de-facto standard for privacy-preserving learning from large datasets with individual records. However, important research and engineering issues remain open to more accurate and efficient solutions. Existing approaches either require large amounts of privacy budget to train accurately, and incur high overheads in terms of computational and memory resources consumption. In this article, we provide a critical review of some of the most prominent DP-SGD approaches, and discuss their relative strengths and weaknesses. Specifically, we look at several adaptive approaches that attempt to improve the privacy-accuracy trade-off during learning, by dynamically adjusting parameters such as clipping threshold, learning rate or budget allocation. We also provide an overview of important data management aspects in DP-SGD, which influence significantly its computational and memory overhead. Finally, we review a novel approach to support heterogeneous privacy requirements, i.e., individualized privacy settings.

1 Introduction

The past decade witnessed the emergence of machine learning, and in particular neural networks, as the tool of choice for a broad area of application domains, e.g., healthcare, social media, computer vision, cybersecurity, etc. For neural network to perform accurately, large input datasets are required to train them. Often, these datasets consist of individuals' data, e.g., user profiles and purchasing records, patient records, user browsing history, etc., which if not carefully handled, may disclose sensitive data regarding one's health status or personal lifestyle details.

To address individual privacy concerns, differential privacy (DP) [1] emerged as the de-facto standard in data protection. The main strength of DP is that it provides formal protection guarantees, and it allows one to derive statistically-significant patterns from large amounts of user-centric data, without allowing an adversary to determine whether the data of any targeted individual has been used or not in the model's training set. In the context of training neural networks, DP has been implemented in conjunction with the popular Stochastic Gradient Descent (SGD) approach, resulting in its privacy-preserving version DP-SGD [2]. While DP-SGD is a powerful tool, it still presents challenges with respect to the trade-off between privacy and utility of learned models, as well as a number of performance concerns. A significant number of research contributions [3, 4, 5, 6, 7],

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

focused on how to adapt DP-SGD and tune it such that utility is increased. In this paper, we provide a review of some of the most prominent adaptive DP-SGD approaches, and we also look at several novel techniques that explore personalized individual requirements [8, 9]. In addition, we explore several data management and performance issues that arise when privately training large models using sizeable datasets. Finally, we provide a brief experimental benchmarking of some of the reviewed approaches, with the purpose of comparing their performance head-to-head on common datasets and under similar parameter settings.

The rest of the article is organized as follows: Section 2 presents fundamental concepts and definitions. Adaptive strategies are overviewed in Section 3. Section 4 focuses on data management and performance issues in DP-SGD. Section 5 looks at approaches for individualized privacy constraints. We perform a comparative experimental analysis of some of the considered approaches in Section 6 to better understand their relative performance, followed by conclusions in Section 7.

2 Background

2.1 Differential Privacy

Dalenius' 1977 principle for statistical databases aimed to ensure individual privacy [10], but achieving it akin to semantic security turned out to be impossible, posing risks even to those not included in the database. More recently, Differential Privacy (DP) emerged as the preferred model for quantifying the added privacy risks for datasets of individual records. The concept of differential privacy was first introduced by Dwork et al. [1], who provided a mathematical framework for quantifying the privacy guarantees provided by a given algorithm.
Differential privacy ensures that attackers cannot discriminate between any two "sibling" inputs by looking at query outputs even when taking into account arbitrary background information. It was proposed in order to offer a statistical guarantee that publicly available aggregate data would not disclose the identity of any individual from the dataset. The exact definition of sibling datasets is provided by the neighboring concept, which in machine learning is best-defined as the neighboring relationship on unstructured data as follows.

Definition 1: *[Neighboring Unstructured Data] Two unstructured datasets $[US]_i$ and $[US]_j$ are said to be neighboring if the Hamming distance between their aggregated real-valued data representations x_i and x_j is at most m [11], where m is the maximum allowed difference, typically set to 1:*

$$d(x_i, x_j) \leq m$$

Assume we have two neighboring datasets, D_1 and D_2, that differ only by a single data item. When we interact with the data via a randomized mechanism M, we say that M is ϵ-differentially private if the probability of observing a given output of M does not vary by more than exp(ϵ) between any two input different datasets, where ϵ is defined as the privacy budget.

Definition 2: *[ϵ-Differential Privacy [11, 12]] A randomized mechanism M satisfies ϵ-differential privacy if for every output $S \subseteq$ range(M) and for all $D_1 \sim D_2 \in \mathcal{D}^n$, the following holds:*

$$\Pr[M(D_1) \in S] \leq \exp(\epsilon) \cdot \Pr[M(D_2) \in S].$$

The indistinguishability constraint has later been relaxed to include a δ additive term, changing the definition to (ϵ, δ)- differentially privacy. The additional parameter allows for a very small probability (bounded by δ) that the algorithm might deviate from the strict privacy guarantee of ε, providing a trade-off between privacy and accuracy.

Definition 3: *[(ϵ, δ)-Differential Privacy [13, 4]] A randomized mechanism M satisfies (ϵ, δ)-differential privacy if for every output $S \subseteq range(M)$ and for all $D_1 \sim D_2 \in \mathcal{D}^n$, the following holds:*

$$\Pr[M(D_1) \in S] \leq \exp(\epsilon) \cdot \Pr[M(D_2) \in S] + \delta.$$

The Gaussian mechanism is a noise-addition mechanism that satisfies (ϵ, δ)-DP (for $\delta > 0$). Through this approach, the L_2 sensitivity of the query function is adjusted with Gaussian noise [13].

Definition 4: *[L_2 Sensitivity [13]] Let $f : \mathcal{X}^n \to \mathbb{R}^d$ be a query function. The L_2 sensitivity of f, denoted by $\Delta_2(f)$, is defined as:*

$$\Delta_2(f) = \max_{X \sim X'} \|f(X) - f(X')\|_2$$

Definition 5: *[Gaussian Mechanism [13]] For any arbitrary ϵ from the interval $(0, 1)$ and a query function q with L_2 sensitivity $\Delta_2(q)$, the Gaussian Mechanism ensures (ϵ, δ)-differential privacy. This mechanism adds Gaussian noise $N(0, \sigma^2)$ to $q(D)$, where σ satisfies*

$$\sigma \geq \frac{\Delta_2(q)}{\epsilon} \sqrt{2 \ln \left(\frac{1.25}{\delta} \right)}$$

Another variation of the differential privacy definition is Rényi Differential Privacy (RDP) [14]. A generalization of the Kullback-Leibler divergence [15], Rényi divergence serves as the foundation for RDP. Conventional DP quantifies the privacy guarantee by means of the privacy loss parameter ϵ, which limits the probability ratio of a mechanism's outputs in the presence of two adjacent datasets. Conversely, RDP offers a more flexible and fine-grained measure of privacy by including a parameter, α (the order of Rényi divergence), in addition to ϵ.

Definition 6: *[Rényi Divergence [4, 14]] For two probability distributions P and Q and a parameter $\alpha > 1$, the Rényi divergence of order α is defined as:*

$$D_\alpha(P \| Q) = \frac{1}{\alpha - 1} \log \left(\sum_x P(x)^\alpha Q(x)^{1-\alpha} \right)$$

Definition 7: *[Rényi Differential Privacy [4, 14]] A randomized mechanism $\mathcal{M}$ is said to be (α, ϵ)-RDP if for any two neighboring datasets D_1 and D_2:*

$$D_\alpha(\mathcal{M}(D_1) \| \mathcal{M}(D_2)) \leq \epsilon$$

In differential privacy, tracking the cumulative privacy loss, or privacy budget, is crucial, particularly in scenarios involving multiple applications or complex compositions of privacy mechanisms. Traditional methods often struggle to provide tight and accurate bounds on this cumulative privacy loss. To address this, a sophisticated method known as the Moments Accountant (MA) has been developed. The MA leverages the framework of Rényi Differential Privacy (RDP) to achieve precise tracking of the privacy budget. By calculating the moments, or expected values of powers, of the privacy loss random variable, it quantifies the degradation of privacy guarantees over successive computations. This method utilizes RDP to evaluate the Rényi divergence between the output distributions of the mechanism for neighboring datasets at each step, allowing for a more refined and accurate assessment of the privacy-utility trade-off.

2.2 Differentially Private Stochastic Gradient Descent (DP-SGD)

Stochastic Gradient Descent (SGD) is a popular optimization technique employed in the training of machine learning models. SGD iteratively adjusts model parameters by leveraging the gradients of a loss function, which are computed based on a randomly selected subset of the training data, known as a batch [16, 17]. The specific rule for updating the model parameters is given by:

$$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t; x^{(i)}, y^{(i)})$$

where θ_t denotes the model parameters at iteration t, η is the learning rate, and $J(\theta_t; x^{(i)}, y^{(i)})$ represents the loss function evaluated on the current batch $(x^{(i)}, y^{(i)})$.

DP-SGD extends Stochastic Gradient Descent (SGD) to provide differential privacy guarantees while training machine learning models on sensitive data. In DP-SGD, gradient clipping is performed before adding noise to the gradients to ensure that the gradients remain bounded.

The clipped gradient $\hat{\nabla} f(\theta, D_t)$ for one sample of data D_t is computed as:

$$\hat{\nabla} f(\theta, D_t) = \text{Clip}(\nabla f(\theta, D_t), C)$$

where:

- $\nabla f(\theta, D_t)$ is the gradient of the loss function with respect to the model parameters θ computed on data sample D_t,
- C is the clipping threshold, which limits the magnitude of the gradients.

Clipping plays a pivotal role in the DP-SGD process in balancing the trade-off between privacy and utility. Clipping the gradients is needed to prevent outliers (extreme or unusually large gradients) and it limits their sensitivity, preventing the model from learning more than a set quantity from any given sample, which prevents inadvertent leakage of information during training and thus satisfies differential privacy. It is important to note that the way clipping is performed can vary, and the method of clipping can end up affecting the algorithms performance. In the original implementation of DP-SGD done by Abadi et. al. [2], a random sample from a mini-batch is used to compute gradients. For each sample gradient, a check is performed to see if the magnitude of the gradient is larger than the clipping threshold. If the magnitude of the gradient is determined to be greater than the clipping threshold, it gets scaled down to its L_2 norm C as follows:

$$\bar{g}_t \leftarrow g_t(x_i)/\max(1, \frac{||g_t(x_i)||_2}{C})$$

where $\max(1, \frac{||g_t(x_i)||_2}{C})$ is the maximum value between 1 and the magnitude of the gradient divided by the clipping threshold. If the magnitude of the gradient is less than the clipping threshold, the gradient would end up being divided by 1, thus preserving the original gradient. If the gradient is larger than the clipping threshold C, the value of $||g_t(x_i)||_2$ will ends up being greater than 1, and the original gradient is divides to scale it down to the clipping threshold.

After clipping the gradients, noise is added to ensure differential privacy:

$$\tilde{\nabla} f(\theta, D_t) = \hat{\nabla} f(\theta, D_t) + \mathcal{N}(0, \sigma C I_n)$$

where:

- $\mathcal{N}(0, \sigma C I_n)$ is the noise term drawn from the Gaussian distribution with mean 0 and scale parameter $\sigma C I_n$,
- ϵ is the privacy parameter, controlling the privacy strength.

After adding noise to the clipped gradients, the model parameters are updated using the noisy gradients:

$$\theta_{t+1} = \theta_t - \eta \tilde{\nabla} f(\theta, D_t)$$

The use of privacy budgets is a crucial component of DP-SGD. Because SGD is iterative, if one applies the traditional sequential composition theorem [18], which states that the total budget for the learning process is equal to the sum of the budgets used in each iteration, the budget consumption can increase significantly. The moments accountant (MA) introduced in [2] offers a tighter bound on privacy budget consumption using Poisson sampling.

The trade-off between privacy and utility in DP-SGD encapsulates the delicate balance between preserving the privacy of sensitive data and maintaining the effectiveness of the trained machine learning model. DP-SGD introduces noise to the gradients during the training process to ensure that individual data points do not unduly influence the model updates, thereby providing strong privacy guarantees. However, this noise addition can degrade the quality of the learned model, leading to a reduction in its performance on the task at hand. Figures 1 and 2 illustrate the difference in the algorithm structure between SGD and DP-SGD.

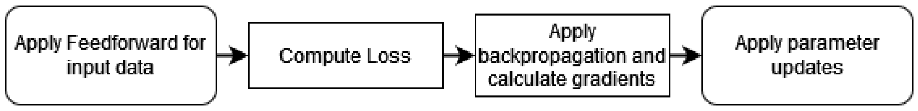

Apply Feedforward for input data
Compute Loss
Apply backpropagation and calculate gradients
Apply parameter updates

Figure 1: Conventional (non-private) SGD workflow

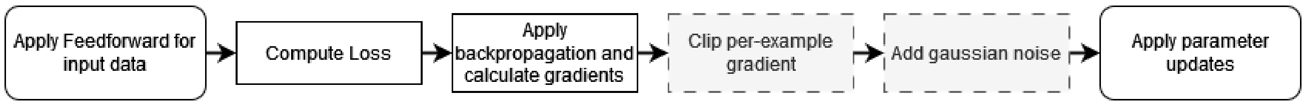

Apply Feedforward for input data
Compute Loss
Apply backpropagation and calculate gradients
Clip per-example gradient
Add gaussian noise
Apply parameter updates

Figure 2: DP-SGD workflow

3 Adaptive DP-SGD Approaches

The performance of ML algorithms is highly-dependent on the properties of the data. In the case of private learning, the constraints imposed by DP create additional challenges, and hence having fixed parameter settings is even less likely to perform well, leading to low accuracy or high privacy budget consumption (i.e., low protection). A significant amount of research explored the idea of adapting parameter values to the training data, or to other inputs such as number of iterations, batch size, privacy budget, learning rate, etc. [5, 7, 19].

In this section, we review several adaptive DP-SGD approaches, which fall mainly into three categories: tuning the privacy budget over time, varying the clipping threshold, or changing the learning rate. Adjusting these parameters offers several advantages. For instance, once can directly control the privacy budget consumption, which in DP-SGD is an output of the noise injection procedure (whereas the input is represented by noise magnitude). Furthermore, the privacy/accuracy trade-off of DP-SGD [20] can be better controlled, as hyperparameter values can significantly affect the performance of trained models [6]. However, in the case of private learning, selecting optimal hyperparameter values is a lot more challenging than in non-private learning, due to the fact that any information used in tuning them has to be itself sanitized, potentially leading to additional privacy budget consumption. In contrast, with non-private learning, one can rely on trial-and-error approaches to do an exhaustive search of the hyperparameter space. Although some existing works have explored the problem or private hyperparameter tuning, they typically do not guarantee optimality of parameters, and they often increase the privacy budget consumed [21].

3.1 Privacy Budget Adaptation

Perhaps the most challenging task in DP-SGD is striking a good trade-off between privacy and utility. DP-SGD injects noise into the gradient values during the training process to ensure that individual data points do not significantly influence the model updates in a way that allows re-identification. However, the noise addition process also degrades the quality of the learned model, leading to a reduction in its performance on the task at hand [2]. Hence, a delicate balance must be achieved between preserving the privacy of sensitive data and maintaining the effectiveness of the trained machine learning model. Next, we review three directions in which this trade-off can be controlled.

3.1.1 Noise Decay

The first study that addressed adapting the DP-SGD privacy budget [20] proposed a solution based on the idea that, as the model converges, the gradient is expected to have a lower magnitude, thus allowing the learning process to converge faster to a local optimal, and achieve higher accuracy. To this end, a set of methods for privacy budget allocation were defined in [20] that dynamically reduce the noise scale as the training time increases.

- **Adaptive Schedule Based on Public Validation Dataset:** As mentioned earlier, one challenging aspect when making data-dependent decisions with private algorithms is that one must consume privacy budget when accessing any data-derived intermediate results. When public datasets are available, this challenge is averted, as one does not need to protect the inputs of the parameter tuning strategy algorithm.

  The main idea in this approach is to continuously check the validation error on the public dataset while training on the private one, and dynamically reduce the noise scale whenever the validation accuracy improves by less than a set threshold δ. In such cases, the noise scale is reduced by a factor of k and this continues until the total privacy budget is consumed. The evaluation intervals where validation is carried out are termed validation epochs.

  Let σ_e denote the noise scale for the DP-SGD training in validation epoch e, and S_e represent the corresponding validation accuracy. The adjustment of the noise scale for subsequent epochs is contingent upon the discrepancy in accuracy between the current epoch e and the preceding validation epoch $e-1$. Initially, $S_0 = 0$.

  The formula for adjusting the noise scale is:

$$\sigma_e = \begin{cases} k\sigma_e, & \text{if } |S - S_{e-1}| \leq \delta \\ \sigma_e, & \text{otherwise} \end{cases}$$

  This adjustment ensures that the noise scale adapts based on the performance change observed between validation epochs. If the accuracy difference $S_e - S_{e-1}$ is less than the predetermined threshold δ, the noise scale is attenuated by a factor of k $(0 < k < 1)$. To improve training effectiveness, the moving average of the validation accuracy is considered in the decision process, as there are cases where the validation accuracy does not increase monotonically within the training progress, and any fluctuations may result in unnecessary reduction of noise scale, which in turn consumes privacy budget.

- **Pre-defined Schedules:** In cases where a public validation dataset is not available, an alternative solution proposed in [20] calculates a fixed, data-independent schedule according to which the noise scale decreases over time. Four such decay strategies are presented:

  a) Time-Based Decay: the noise scale is adjusted using the formula $\sigma_t = \sigma_0/(1 + kt)$, where σ_0 is the initial noise scale, t is the number of training epochs so far, and $k > 0$ is the decay rate. When $k < 1$, this method is referred to as "search-then-converge", and the noise scale decreases linearly during the search phase when t is less than the "search time" $1/k$; afterwards, the noise scale decreases by a factor of $1/t$.

b) Exponential Decay: The noise scale decreases exponentially with each epoch, according to the expression $\sigma_t = \sigma_0 e^{-kt}$, where $k > 0$ is the decay rate.

c) Step Decay: The noise scale decreases by an exponentially-increasing factor every few epochs, according to expression $\sigma_t = \sigma_0 * k^{t/period}$. The decay rate k is chosen such that $0 < k < 1$.

d) Polynomial Decay: The noise parameter follows a polynomial decay function over a specified number of epochs $period$. Specifically, $\sigma_t = (\sigma_0 - \sigma_{end}) * (1 - t/period)^k + \sigma_{\text{end}}$ where $k > 0$ is the decay rate and $t < period$. When $k = 1$, this is referred to as a linear decay function.

3.1.2 Optimal Step Size Search

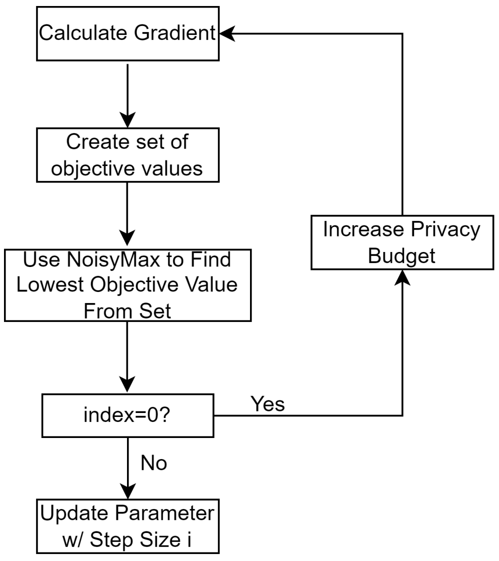
The previous approaches looked solely at how to adapt the privacy budget allocation throughout the learning process using different decay functions over time. Next, we look at an approach [13] that adapts both the privacy budget and the learning rate (while we dedicate a separate section to learning rate adaptation approaches in Section 3.3, we discuss this hybrid method here). At the core of the proposed technique from [13] sits a privacy-preserving algorithm for computing the noisy maximum among the values in a set. The NoisyMax algorithm adds independent Laplace noise to each set value and returns the index of the largest one, thereby providing differential privacy guarantees [13].

The two main components of the algorithms are step-size selection and adaptive noise reduction. The goal of step-size selection is to efficiently choose the per-iteration privacy budget. Adding noise to the gradients may not guarantee the correct direction of the descent, in expectation. To alleviate this issue, a portion of the privacy budget p_{nmax} is used to check whether a given noisy estimate $\tilde{g}_t$ of the gradient gives the correct descent direction. To accomplish this, a set $\Omega = \{f(w_t - \alpha\tilde{g}_t) : \alpha \in \Phi\}$ is constructed, where each element of the set is the objective value evaluated at $f(w_t - \alpha\tilde{g}_t)$ and Φ is a set of pre-defined step-sizes. Using the NoisyMax computation, the algorithm calculates which step-size results in the smallest objective function value. Since an a priori bound cannot be determined on a given loss function l, gradient clipping is applied to bound sensitivity. This way, the first element of ϕ is fixed to 0, to take into consideration the current objective value. Let i be the index returned by the NoisyMax algorithm. If $i > 0$, the algorithm updates w_t using the chosen step size a_i. However, if $i = 0$, $-\tilde{g}_t$, it is likely that is not the correct descent direction, and none of the given step sizes will lead to a decrease in the value of the objective function f. Hence, the adaptive noise reduction is applied.

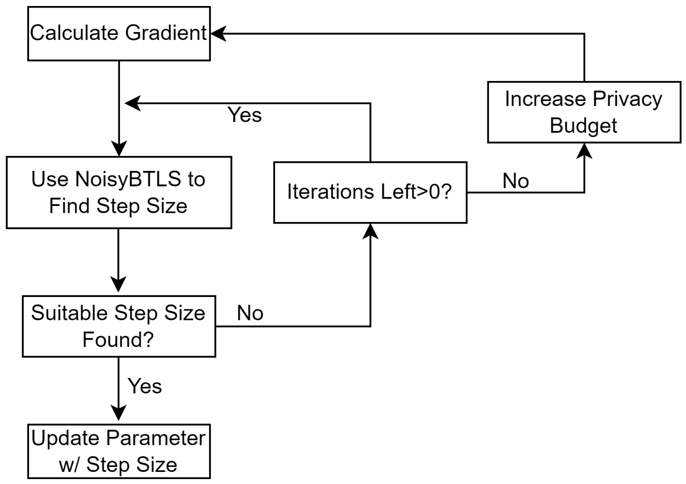
A block diagram of the NoisyMax technique is illustrated in Figure 3. When the NoisyMax algorithm determines that the current gradient direction is incorrect, the algorithm increases the privacy budget used in noisy gradient approximation by a factor of $1 + y$. Using the gradient averaging technique, a total privacy budget of $p_{\text{ng}} \breve{} p_{\text{old}}$ is consumed to measure the new gradient with increased accuracy. The gradient averaging technique is a method that recycles gradient estimates that were not useful by using the difference between two related privacy budget values to measure a new gradient, which when combined with the old measured gradient can be used to measure the final noisy gradient estimate at a lower privacy budget. Once the algorithm measures a new gradient, it goes through the NoisyMax algorithm again to determine its direction. These steps repeat until a good descent direction can be found.

As an extension to NoisyMax, the same authors proposed a more advanced approach called Noisy Backtracking Line Search (NoisyBTLS). Similar to NoisyMax, NoisyBTLS adapts both the learning rate and the privacy budget, but in addition it also considers adaptive clipping (which is covered in more detail in Section 3.2).

This method focuses on adapting the learning rate in response to noisy gradient computation while also adjusting the privacy budget using Sparse Vector Technique (also known as the "above-threshold" mechanism) [22, 23]. The approach first conducts a backtracking line search in a differentially private manner. Figure 3(b) illustrates


Calculate Gradient
Create set of objective values
Use NoisyMax to Find Lowest Objective Value From Set
index=0?
Yes
No
Update Parameter w/ Step Size i
Increase Privacy Budget

(a) NoisyMax


Calculate Gradient
Yes
Use NoisyBTLS to Find Step Size
Iterations Left>0?
No
Increase Privacy Budget
Suitable Step Size Found?
No
Yes
Update Parameter w/ Step Size

(b) NoisyBTLS

Figure 3: (a) NoisyMax Technique; (b) NoisyBTLS Technique

this approach. The algorithm starts by introducing noise to the threshold $T = 0$, resulting in a noisy threshold $T = \lambda$, where λ is randomly drawn from a Laplace distribution or alternatively, a Gaussian distribution. Next, multiple iterations are performed, and in each one the following queries are evaluated:

$$q(\eta, D) = f(w) - f(w - \eta\nabla f(w)) - \alpha\eta\|\nabla f(w)\|^2$$

with added noise ν at each iteration. The result is compared with the noisy threshold T. If $q(\eta, D) + \nu \geq T$, the algorithm outputs η and halts; otherwise, it reduces the step size η by multiplying it with β and continues to the next iteration, where $\beta \in (0, 1)$ controls the rate of step size reduction. This process repeats for a maximum number of iterations. If no limit is set, depending on the noise used in the query, the iteration could lead to an infinite loop or provide minuscule step sizes, which could end up increasing the objective function due to a lack of progress. When the maximum number of iterations is reached, it statistically calculates whether a higher privacy budget is needed, which it then adjusts depending on the results. Due to the use of SVT, the privacy budget needed to find η is greatly reduced.

When NOISYBTLS fails to find a step size within the set maximum iteration count, it can lead to two possibilities. The first one is that the current privacy budget p_{grad} is set too small, and the noise dominates the gradient – Case (1) in the diagram. In this case, the privacy budget needs to be increased. The second possibility is that the noise used in NOISYBTLS is too large and it can't identify a step size with the right conditions – Case (2) in the diagram. The remedy to this problem is to increase the privacy budget in order to compute a more precise gradient. To identify which of these two cases are applicable, the algorithm maintains the moving average of angles between two consecutive gradient values, which gets updated at every iteration:

$$\theta \leftarrow \text{ANGLE BETWEEN}(g_t, \tilde{g}_t)$$

$$\theta = \psi\theta + (1 - \psi)\theta_{t-1}$$

where $\psi \in (0, 1)$ is a parameter controlling the decay rate of old information. When $\eta = 0$ is returned by NOISYBTLS, the algorithm calculates another gradient $\tilde{g}_{t2}$ using the budget of ρ_{grad} and measures the angle θ between $\tilde{g}_t$ and $\tilde{g}_{t2}$). The value of θ_{t2} is then compared with the moving average-based threshold θ_{max}, and if it is larger it would increase the privacy budget $_{\text{grad}}$ for calculating the gradient. Meanwhile, if θ is less than the minimum threshold θ_{min}, the search might fail due to insufficient privacy budget ϵ_{BT} for the NOISYBTLS. The threshold values θ_{max} and θ_{min} are calculated as follows:

$$\theta_{\text{max}} = \phi_{\text{max}} \times \theta, \quad \theta_{\text{min}} = \phi_{\text{min}} \times \theta$$

where $\phi_{\text{max}} > 1$ and $0 < \phi_{\text{min}} < 1$ are hyperparameters. Empirically, this budget adaptation strategy is observed to be particularly effective for convex optimization problems.

In addition to adjusting the privacy budget, it was found that a large clipping threshold is not necessary in the later stages of the training process. Knowing this, the clipping threshold C_{grad} and C_{obj} is adaptively decreased when the algorithm finds it needs to increase the privacy budget p_{grad} during a single SGD update. Even if p_{grad} is increased multiple times in a single SGD update, the clipping threshold will only be updated once per update. The clipping threshold is updated as follows:

$$C_{\text{grad}} \leftarrow (1-\zeta)C_{\text{grad}}, \quad C_{\text{obj}} \leftarrow (1-\zeta)C_{\text{grad}}$$

where ζ is a hyperparameter that determines the rate of decrease. Since the condition is based on privately released information, it does not consume any extra privacy budget.

3.1.3 Convergence Rate

Another recent approach to adapting the privacy budget consumption relies on convergence rates [24]. To assess the effectiveness of Private Gradient Descent (PGD), this work utilizes the Expected Excess Risk (EER), a common metric for evaluating the convergence of randomized algorithms. Given the presence of noise and the constraint on learning iterations, optimization using private gradients is expected to lead to a higher loss (excess risk) compared to the optimal solution without privacy constraints. Let θ^* be the optimal solution obtained after iterating an algorithm for T times. EER quantifies the expected utility degradation as:

$$\text{EER} = \mathbb{E}[f(\theta_{\nu}^{T+1})] - f(\theta^*).$$

Due to the variety of loss functions and complexity of recursive iterations, a good estimation of EER in the presence of noise is intractable for most functions. Instead, one can study the worst-case scenario, i.e., the upper bound of EER, with the goal to minimize the upper bound. For consistency, we refer to the upper bound of EER divided by the initial error as $ERUB$. Since the analytical form of EER is either intractable or complicated due to the recursive iterations of noise, studying $ERUB$ is a convenient and tractable alternative. The upper bound often has convenient functional forms which are (1) sufficiently simple, so that they can be directly minimized, and (2) closely related to the landscape of the objective depending on both the training dataset and the loss function. As a consequence, it was used in previous literature for choosing hyperparameters. Denote by $ERUB_{\text{min}}$ the achievable optimal upper bound by a specific choice of parameters, e.g., noise magnitude σ and T.

One can define the influence of noise magnitude σ on EER as the derivative:

$$q_t^* = \frac{\partial \text{EER}}{\partial \sigma_t}.$$

Accordingly, one can approximate the EER shift as $\Delta\sigma$ when σ increases by $\Delta\sigma$. However, because the EER is strongly data-dependent, the derived q_t^* on a given dataset may not generalize to another dataset. Instead, one can consider a more general term based on $ERUB$, i.e., $q_t^* = \frac{\partial}{\partial ERUB}\sigma_t$.

The work in [24] considers the class of loss functions satisfying the Polyak-Lojasiewicz (PL) condition, which bounds losses by their corresponding gradient norms. It is more general than the m-strongly convexity condition [25]. If f is differentiable and M-smooth, then m-strongly convexity implies the PL condition.

The related method in [19] introduces a conceptual framework aimed at dynamically adjusting hyperparameters over time by optimizing the privacy-utility ratio (PUR) at each step. This involves selecting the step size η and noise standard deviation σ to minimize the privacy loss per unit of utility improvement. The PUR is defined as the ratio of the privacy cost to the utility gain, with utility incorporating convergence metrics like the gradient norm or objective value. Minimizing the PUR enables adaptation of the privacy budget based on optimization progress, with higher precision typically required in later stages as the gradient norm diminishes near the optimum.

Two selection strategies emerge:

1. Data-dependent selection: By assuming F is M-smooth, a descent lemma estimates the expected improvement in the objective function given step-size and noise variance. Using this lower bound as utility function, along with the privacy cost, one determines hyperparameters that minimize the privacy-utility ratio. Specifically, the privacy-utility ratio is minimized by setting the noise standard deviation σ_t proportional to the gradient norm and the step size η_t as a constant, given by:

$$\sigma_t = \frac{\|\nabla F(\theta_t)\|}{\sqrt{d}}$$

$$\eta_t = \frac{1}{2M}$$

2. Data-independent selection: A data-independent schedule can be derived based on the PUR-optimal schedule, which is proportional to the gradient norm. This schedule exhibits similar convergence rates to non-private gradient descent (GD), leveraging upper bounds on gradient norms as a proxy for the gradient norm itself.

3.2 Gradient Clipping Adaptation

The clipping threshold parameter is used to bound the sensitivity of each gradient. A low clipping parameter can result in information being destroyed, and may change the direction of the gradient step. Meanwhile, a high clipping bound increases the sensitivity of training and requires more noise to be added [2]. It is difficult to achieve a good fixed clipping parameter setting without looking at the training data. Furthermore, weight layers and bias layers need completely different clipping values to be optimal. To tackle this issue, several approaches proposed their own implementation of adaptive clipping. In Section 3.1.2 we briefly discussed one method that pairs adaptive clipping with other adaptive parameters. One important aspect is that in a private setup, using gradient norms for tuning requires them to be sanitized first, which means that a portion of the privacy budget must be allocated in each step to protect the norms [6].

3.2.1 Norm-based Adaptive Clipping

One of the first proposals of adaptive clipping was introduced by Van der Veen et.al. [26] who recognized that choosing a good clipping threshold C is difficult, especially when dealing with multiple layers. They proposed that the clipping bound in the current batch be directly proportional to the l_2-norm of the previous batch by a constant factor α. This can be summarized in the equation below:

$$C_t^l = \alpha |L|^{-1} (\sum_{i \in L_t} \text{clip}(||g_{t-1}^l(x_i)||_2) + \mathcal{N}(0, \sigma_{l^2}^2 {C_{l^2 t}^l}^2)$$

$$\text{clip}(||y||_2) = ||y||_2 / \text{max}(1, \frac{||y||_2}{C_{l^2 t}^l})$$

where C_t^l is the clipping bound of the current round t and layer l, $g_{t-1}^l(x_i)$ equates to the individual gradient of that layer in previous rounds and privacy parameters σ_{l_2} and $C_{l^2 t}^l$. To calculate the privacy parameter $C_{l^2 t}^l$, an adaptive procedure is performed similar to clipping, where they sanitize the l_2-norm of the previous iteration, C^l_{t-1}, multiplied by a constant β, hence $C^l_{l^2 t} = \beta C^l_{t-1}$. In this approach, it is required for the user to manually set the values of α, β, and σ_{l2}, although it has been shown that changing the values of β and σ_{l2} does not provide meaningful results when measuring test accuracy.

3.2.2 Quantile Estimation Adaptive Clipping

The recent work of He et. al. [6] explores an adaptive clipping technique that uses a form of quantile estimation in the context of per-layer clipping. A portion of the total privacy budget ($r = 1\%$ to 10% of total budget) is allocated for estimating the target quantile for each layer's gradient norms. The clipping threshold for each layer $C_1, ..., C_K$ is then set to the estimated quantile. The number of gradients clipped in each layer is recorded before each parameter update, and the clipping threshold is then adjusted based on how many individual gradients have been clipped previously. The fraction of clipped gradients needs to be sanitized according to DP, and an additional noise multiplier is used to achieve this goal. This also affects the noise multiplier setting for parameter updates, and the new noise multiplier is calculated as:

$$\sigma_{\text{new}} = (\sigma^{-2} - K(2\sigma_b)^2)^{-1/2}$$

where σ_b is the additional noise multiplier for the sanitized clipped gradients fraction, σ is the original noise multiplier and K is the number of layers in the neural network (also known as number of groups).

The technique also uses different levels of noise in each layer. For example, let $\gamma_1, ..., \gamma_k$ be coefficients for scaling, and $\tilde{g}_k$ be the sum of clipped gradients for layer/group k. Applying the Gaussian mechanism to scaled $\hat{g} := (\hat{g}_1, ..., \hat{g}_k)$, where $\hat{g}_k := \tilde{g}_k/\gamma_k$, and rescaling back the sanitized values afterwards adds noise to $\tilde{g}_k$ with standard deviation proportional to γ_k. There are two ways to choose $(\gamma_1, ..., \gamma_K)$:

- Global strategy: $\gamma_k = 1$ for $k \in [K]$. This strategy adds same amount of noise to all components.
- Equal budget strategy: $\gamma_k = C_k$ for $k \in [K]$. This strategy gives all the groups the same privacy budget.

Another method of adaptive gradient clipping has been explored in [3], this time in the context of federated learning (FL). The main idea of the approach is to fix the quantile value of observed norms, and use gradient descent to fit C to this value:

Let $X \in \mathbb{R}$ be a random variable, and let $\gamma \in [0, 1]$ be a quantile to be matched. For any C, define

$$l_\gamma(C, X) = \begin{cases} (1-\gamma)(C - X), & \text{if } X \le C \\ \gamma(X - C), & \text{otherwise} \end{cases}$$

which implies

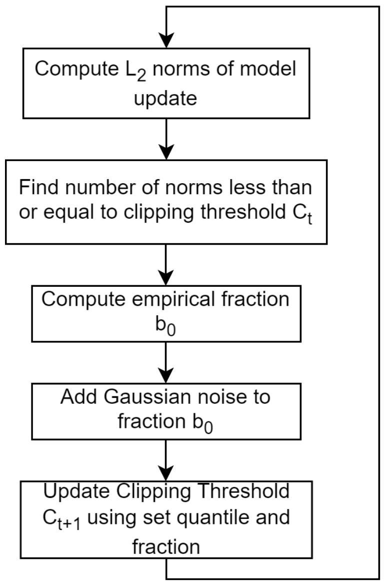
$$\nabla l'_y(C, X) = \begin{cases} (1-\gamma)(C - X), & \text{if } X \le C \\ -\gamma, & \text{otherwise} \end{cases}$$

Since the loss is minimum when $Pr(X < C) = \gamma$, the loss function is convex and gradients are bounded by 1, it is possible to get an online estimate of C that converges to the γ^{th} quantile of X using online gradient descent. On a sample size m, with b being the proportions of elements lower than C, the average derivative of the loss for that round can be simplified to $\bar{b} - \gamma$. This is captured by the following equation:

$$\bar{l'}_\gamma(C; X) = \frac{1}{m} \sum_{i-1}^{m} \begin{cases} (1-\gamma), & \text{if } x_i \le C \\ -\gamma, & \text{otherwise} \end{cases}$$

$$= \frac{1}{m}((1-\gamma)\sum_{i\in[m]} \mathbf{I}_{x_i} \leq c - \gamma \sum_{i\in[m]} \mathbf{I}_{x_i} > c) = \bar{b} - \gamma$$

For a particular learning rate ηc, the clipping bound can be updated through: $C \leftarrow C - \eta c(\bar{b} - \gamma)$. Since b and γ take values in [0,1] the update clipping bound changes by at most $\eta \times c$ in each step. This can be a problem in two scenarios, one if C is very large, and the other when updates are coarse and may overshoot to become negative if C is a lot smaller than ηc. The following geometric update rule can be used in this case: $C \leftarrow C \cdot \exp(-\eta c(\bar{b} - \gamma))$, which allows the update rule to quickly converge to the true quantile even if initial estimates are largely different. Figure 4 illustrates the adaptive clipping algorithm using quantiles.



Compute L_2 norms of model update
Find number of norms less than or equal to clipping threshold C_t
Compute empirical fraction b_0
Add Gaussian noise to fraction b_0
Update Clipping Threshold C_{t+1} using set quantile and fraction

Figure 4: Adaptive Clipping Using Quantiles

3.2.3 Coordinate-wise Adaptive Clipping

The work of Pichapati et. al.[5] proposed another approach to adaptive clipping. The AdaClip algorithm adds less noise compared to other methods by using coordinate-wise adaptive clipping of the gradient, as opposed to norm-based, thus producing models with improved accuracy. The main idea in the approach is rather than searching for a good clipping value, the gradients are centered and standardized before being clipped to 1 and then noise is added to their value scaled according to the fixed sensitivity of 1. The noisy values are then transformed back to their original mean and variance.

Denote by g^t the stochastic gradient vector at iteration t, and let a^t and b^t be auxiliary vectors. The gradient vector g^t is first translated by a^t obtaining $(g^t - a^t)$, then each dimension is divided by b^t. This produces the transformed gradient w^t, given by $w^t = \frac{g^t - a^t}{b^t}$. Sensitivity is bounded by clipping the transformed gradient at norm 1, according to equation:

$$\hat{w}^t = \text{clip}(w^t, 1) \triangleq \frac{w^t}{\max(1, ||w^t||_2)}$$

Noise is then added to the clipped gradient:

$$\tilde{w}^t = \hat{w}^t + N^t$$

$$N^t \sim \mathcal{N}(0, \sigma^2 I)$$

Finally, the noisy gradient is rescaled to the same scale as the original gradient by first multiplying with b^t and then adding a^t:

$$\hat{g}^t = b^t \hat{w}^t + a^t$$

This produces the differentially-private approximation of g^t. The main challenge consists in finding the optimal choices of the auxiliary vectors a^t and b^t. When testing the implementation of AdaClip with the original DP-SGD implementation done by Abadi et al [2] on the MNIST dataset, AdaClip was found to produce higher accuracy for the same settings of ε and δ.

3.3 Learning Rate Adaptation

The learning rate parameter controls the step size taken during the optimization process to update the weights of the neural network. Choosing an optimal learning rate is very important: a small learning rate can result in the model taking too long converge, or being stuck at a local optimum. Meanwhile, a learning rate that is too large may result in a model that does not converge.

Adapting the learning rate is a concept that has been used in conventional, non-private machine learning, in tasks involving large-batch training [27]. In the context of DP-SGD, adapting the learning rate is even more important, as clipping and adding noise to gradients can change the direction of the gradient.

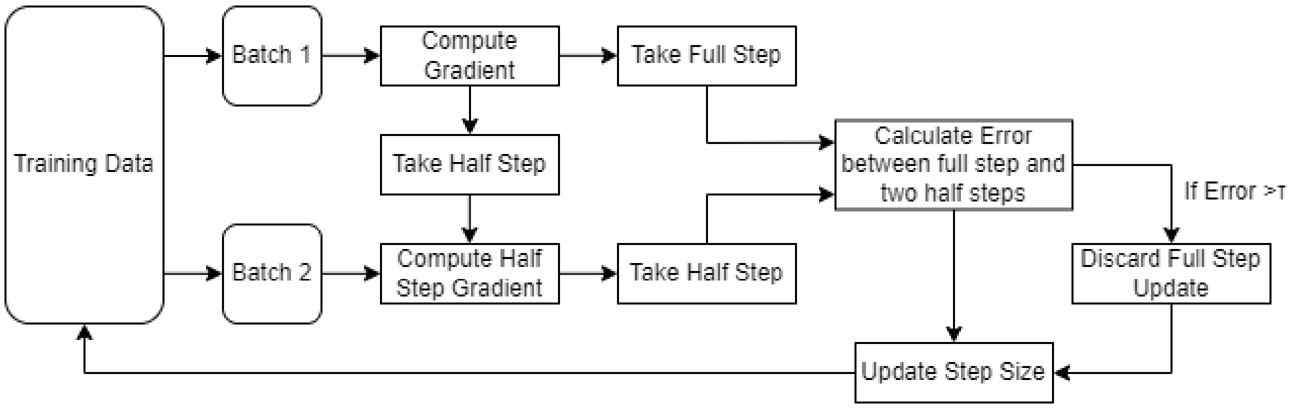


Figure 5: ADADP Update Mechanism

The most prominent approach using adaptive learning rate in private learning has been proposed in [7] by Koskela et al. They proposed a method of adapting the learning rate by estimating the error against the gradient flow when comparing the results after one step and two half-steps. Figure 5 provides an overview of how the algorithm functions, and how the step size is adaptively tuned. The basis of the approach relies on numerical extrapolation of ordinary differential equations (ODEs). Let g be a differentiable function $g : R^d \to R$. To find the local minimum of function g, a first-order method called gradient descent (GD) is used. The gradient flow of g can be described as the explicit Euler method with step size η_l applied to a system of ODEs. An estimation of the error of the gradient descent can be performed by considering one step of size η:

$$\theta_1 = \theta_0 - \eta \nabla g(\theta_0)$$

An alternative approach is to employ two steps of size $\frac{\eta}{2}$:

$$\theta_{1/2} = \theta_0 - \frac{\eta}{2} \nabla g(\theta_0) \qquad \hat{\theta}_1 = \theta_{1/2} - \frac{\eta}{2}(\theta_{1/2})$$

It is thus possible to get an $O(\eta^3)$-accurate estimate of the local error through the value $2(\hat{\theta}_1 - \theta_1)$. Using this information, the estimation of the error at iteration l can be deduced from $\text{err}_l = ||\hat{\theta}_{l+1} - \theta_{l+1}||$. The local error of magnitude τ to be desired is then set, which is used in the following proposed mechanism to update the step size:

$$\eta_{l+1} = \min(\max(\frac{\tau}{\text{err}_l}, \alpha_{\min}), \alpha_{\max}) \cdot \eta_l$$

where the value of $\alpha_{\min} < 1$ and $\alpha_{\max} > 1$. When this mechanism is implemented in a DP-SGD setting, the equation to determine the local error is changed instead to be the estimate of the ℓ_2-norm of the function $\text{err}(\theta, \hat{\theta})$ as it was found to give better numerical results.

$$\text{err}(\theta, \hat{\theta})_i = \frac{|\theta_i - \hat{\theta}_i|}{\max(1, |\theta_i|)}$$

The way the adaptive DP algorithm works is that a random batch B_1 is first drawn with probability $q = |B|/N$. The gradient G_1 is then calculated and clipped by C, before being evaluated at θ_l. The algorithm then performs two different steps, one by step size η_l, $\eta_{l+1} \leftarrow \theta_l - \eta_l G_1$, and the other by half step size, $\frac{\eta_l}{2}$, $\eta_{l+1/2} \leftarrow \theta_l - \frac{\eta_l}{2} G_1$. A second set of batches B_2 with probability q is then drawn, gradient calculated and clipped G_2. This batch, however, is only being evaluated with the half-step size $\eta_{l+1/2}$ performed in the first batch, and another half step is performed, $\hat{\theta}_{l+1} \leftarrow \theta_{l+1/2} - \frac{\eta_l}{2} G_2$. The error between the updates done by one step size, θ_{l+1}, and two half step size, $\hat{\theta}_{l+1}$ is then evaluated using $||\text{err}(\theta_{l+1}, \hat{\theta}_{l+1}||$. If the evaluated error is greater than the set tolerance parameter τ, the updated parameter is discarded. Regardless of whether the updated parameter was kept or discarded, the next iteration step-size η is then calculated using equation (3.3).

One of the main advantages of this algorithm is that its privacy properties are simple to analyze. The complete algorithm can be modeled as $\tilde{M}$ – a composite of two different mechanisms: $M_{G_1}(X)$ and $M_{G_2}(X)$. The SGD approximation as well as the additive Gaussian noises are independent of each other, meanwhile, the sampling ratio q is kept the same for both. By keeping parameters q, σ and C the same for both mechanism, the composite algorithm is able to run half as many iterations as DP-SGD to get the same privacy for the data.

One of the main difficulties in this implementation is choosing a good tolerance parameter τ that allows the accumulated additive noise to stay bounded as well as preventing any instabilities caused by the SGD gradient. This can be achieved by setting τ such that the accumulated DP noise after T iterations is approximately $O(1)$ element-wise.

3.4 Discussion

We explored several categories of techniques for adapting hyperparameters in DP-SGD, all targeting an improvement in the privacy-accuracy trade-off of learning. Table 1 provides a synthetic classification of the different methods covered in this section, and the papers in which they were introduced.

In the category of adapting the privacy budget, the advantage gained is achieved by fine-tuning the injected noise at different stages of learning. In the first few iterations, larger gradients are expected, hence a large privacy budget allocation may not help, but as the parameters approach their optimal values, gradients become smaller and require finer-grained tuning, hence the noise should be reduced [13]. This aspect is further observed in the noise decay approaches, where a simple budget allocation strategy is able to achieve higher accuracy under fewer epochs compared to a uniform privacy budget [20]. However, that's not to say the adaptive privacy budget is a perfect solution. There are usually constraints on where this approach is effective. For example, adapting the privacy budget based on the convergence rate requires making assumptions on the loss function, which is not always possible [19]. Furthermore, some methods require a public validation dataset to be utilized, and when one is not available, the alternative approach requires setting some additional parameters, which can be difficult to optimize [20] and will consume budget otherwise allocated to computing gradients.

With adaptive clipping techniques, the main advantage gained is eliminating the need to fine-tune the clipping threshold parameter during training. We discussed earlier in the section the importance of setting a good clipping

Table 1: Summary Of Adaptive DP-SGD Approaches

Method	**Adaptive Hyperparameters**		
	Privacy Budget	**Clipping**	**Learning Rate**
Noise Decay	[20]		
Optimal Step-Size Search	[13],[4]		[13],[4]
Privacy-Based		[4]	
Convergence Rate	[24],[19]		
Norm-Based		[26]	
Quantile Estimation		[6],[3]	
Coordinate-wise		[5]	
Error Tolerance			[7]

threshold, especially taking into account the sensitivity of the training process. Adaptive clipping helps balance the privacy-utility trade-off to a reasonable degree. In turn, this improves the stability of the training process, preventing extreme cases such as gradient explosions [3]. In some cases, adaptive clipping is necessary to achieve good accuracy results compared to a fixed clipping threshold, as observed with per-layer clipping [6]. It is also important to look at the drawbacks associated with adaptive clipping. Some methods, e.g., quantile estimation, require allocating some privacy budget to perform the necessary calculations [3, 6]. Furthermore, a lot of the techniques used to implement adaptive clipping are dependent on additional hyperparameters in order to work, and depending on the sensitivity of these hyper-parameters the algorithm can influence the performance of adaptive clipping [5]. It is very difficult to choose an optimal value that can provide the best performance for each method.

The last adaptive hyperparameter we discussed is learning rate. The main advantage seen in adaptive learning rate is accelerating the convergence of the model during the training process. Dynamically adjusting the learning rate based on the gradient updates allows the model to make significant progress in a smaller number of iterations [7]. This is especially the case during the early stages of training, when large model updates are beneficial as they accelerate convergence. However, it is important to factor in the additional computational overhead needed to adjust the learning rate at each iteration. This can end up causing the training time to significantly increase. Similarly to other adaptive techniques, some methods of adaptive learning rate are dependent on more hyperparameters that need to be carefully tuned for performance [7]. It is also important to note that the performance of adaptive learning rate is highly dependent on the datasets and tasks being performed [13].

4 Data Management Solutions for DP-SGD

Memory management has always been a critical challenge in machine learning, particularly in the context of private training. A significant part of the difficulty arises from the model itself, especially with the increasing popularity of Large Language Models (LLMs), which can involve billions to trillions of parameters [28]. The use of graphics processing units (GPUs) has become popular for training due to their parallel processing capabilities, high throughput, and specialized hardware optimized for matrix calculations, which plays a pivotal role in machine learning during the forward pass and backpropagation process. Despite these computational advantages, leveraging GPUs to their full potential often presents substantial challenges.

Key issues include the limited memory capacity of GPUs, which restricts the size of models and batches that can be processed [29]. Efficient memory partitioning is necessary to avoid fragmentation and out-of-memory errors but becomes more complex with the additional requirements of DP-SGD, such as gradient clipping and noise addition [2]. Data transfer overheads between CPU memory and GPU memory also pose significant challenges, exacerbated by the frequent updates needed for privacy-preserving operations. Additionally, while multi-GPU training can alleviate some memory constraints, it introduces new complexities in synchronizing gradients and privacy budgets across multiple GPUs.

Understanding and addressing these memory constraints is essential for optimizing the performance and scalability of machine learning models, especially when implementing privacy-preserving algorithms like DP-SGD. A significant challenge in memory management is handling the gradients. Since DP-SGD requires manipulating gradients through clipping and noise addition to preserve privacy, it can significantly influence the memory requirements for training. This section will explore different clipping techniques used in DP-SGD, discussing their key processes and how they impact the memory requirements of training.

4.1 Flat Clipping

4.1.1 Per-example Processing

One significant drawback of DP-SGD is its perceived computational expense, mainly due to the process of clipping per-example gradients. This process, along with the potential normalization of these gradients, imposes substantial memory and time costs within standard machine learning frameworks [30]. As a result, private machine learning techniques like DP-SGD become even more computationally and memory intensive compared to their non-private counterparts [31].

The computational and memory cost of training depends on the technique being used in taking the training data before performing the private training that would effect how the dataset gets clipped in each iteration. The most basic one is per-example clipping, where only one example from the dataset is used in each iteration of the training. Since only one example is being processed at any given time, this can end up easing the memory requirement of the training process. However, it may still be prone to issues in terms of memory management. In particular, these issues can arise from having a large model with billions of parameters, in which the advantage of only processing one example at a time becomes negligible in terms of memory space. In such cases, a solution would be to partition the model into different memory segments, or assign each partition to different GPU units [32]. In the latter case, the overhead introduced in communicating and synchronizing the different GPUs would also need to be considered. Furthermore, this issue can also arise when working with a model that contains multiple layers. The aggregation of the gradient from these layers can end up not fitting in a memory, necessitating to partition the gradients between different memory blocks. Since this issue mainly stems from needing to store the per-example gradient in order to calculate the norm needed for clipping, alternative approaches to clipping can be introduced which do not rely on the gradient norm. This would solve the overall need of storing the per-example gradient in memory. However, it is important to note that per-example clipping tends to require the least memory compared to other clipping techniques as only one example is being worked on at a time. The downside consists mainly in its computational inefficiency, as more iterations are required to go through enough of the training data, and difficulty of implementing parallelism in this method wastes some of the hardware utilization potential.

4.1.2 Minibatch Clipping

Mini-batching processes a subset of examples from the dataset (a minibatch) at the same time for each iteration. Compared to per-example clipping, minibatching has the advantage of being more computationally efficient, as less iterations are needed to process the same number of examples. Hardware utilization is also more efficient due to allowing parallelization and vectorized operations [33]. However, it is widely acknowledged that minibatch clipping will require more memory usage compared to other techniques. Not only do all the examples need to be stored, but the related activation function result and gradient would end up needing to be stored to perform the clipping. As seen for per-example clipping, if the model itself ends up taking a large amount of memory space, this problem would be exorbitantly worse in the minibatch setting. The overall memory size requirement can be dictated by the batch size multiplied by the size of each example in the batch, meaning the bigger the batch size, the more memory is required [34]. Additional temporary memory buffers must be allocated for the aggregation of the gradients in the minibatch before noise can be added and the parameters can be properly updated. However, if small batches are used to reduce memory overhead, it would result in lower computational performance as it limits the capabilities of parallelization [35].

To ease the memory requirement, the aggregation of the gradients can be done from the start, eliminating the need to store the per-example gradients of the batch, but realistically this is applicable only for clipping techniques that do not rely on the gradient norm. Alternatively, the idea of gradient accumulation can be implemented to reduce the memory needed [36]. The concept works similarly to microbatching, which is explained later. Similar to previous steps, a small batch is taken and the gradients in the batch are clipped and aggregated into one memory block. The next batch is then drawn and gradients of the batch are clipped again, but this time instead of storing them separately from the first batch, they would all be accumulated in one running sum. This repeats for a number of steps before performing the gradient step with noise addition to do the parameter updates. This technique allows for an arbitrarily large batch while requiring a fixed amount of memory.

4.1.3 Microbatch Clipping

Microbatching takes mini-batching further, by dividing the minibatches to even smaller sub-batches called micro-batches. This technique eases up the memory requirements of minibatching, by working with fewer inputs at any given time. Assume a minibatch with a size of 32, meaning that 32 activations and gradients need to be stored simultaneously. Assume we introduce microbatching and split the minibatches into 4 microbatches of size 8 each, we would then only need to hold 8 instances of activation and gradients at the same time. Once the current microbatch has finished, only the final aggregated gradient of the microbatch needs to be stored, and the memory to store the previous gradients would then be used for the next iteration. These microbatch gradients would all be accumulated in one memory block that stores the overall aggregated gradient of the microbatch. Figure 6 shows a comparison between the differences of minibatch clipping and microbatch clipping. One of the main advantages is the ability to add noise to each micro-batch as opposed to the aggregated gradient in mini-batching, allowing tighter control over the DP mechanism and allowing more precise tuning to the privacy required [36]. However, this additional noise can also result in reduction in the model's utility. Microbatching also results in higher runtime compared to traditional minibatching, as it limits the capabilities of the hardware utilization compared to traditional minibatching due to the need to process the micro-batch sequentially [34]. It may seem counter-intuitive to use microbatching, given that minibatching offers greater computational advantage and there is the option to simply reduce the batch size to handle the memory problem, but in some cases microbatching is the only alternative solution that is available. For example, a single GPU may not be sufficient to support large batch sizes. By utilizing micro batching, the peak memory required is significantly reduced, allowing the handling of larger batch sizes with limited resources [37].

To combat the high memory and computational demands of DP-SGD, several works investigated solutions that ease the cost of DP-SGD, especially in the contest of large-scale models. McMahan et al. [38] modified federated learning techniques for DP-SGD and distributed the training process to different mobile devices that share the same model. Dupuy et al. [31] proposed another technique for group-wise clipping involving the use of GPUs. Since DP-SGD is computationally expensive because each batch requires its own gradients, these batches can instead be divided into micro-batches and assigned their own GPU. This reduces the computational cost of computing the overall gradient as the work is divided between the GPUs.

4.2 Group-wise Clipping

He et al. [6] investigated group-wise clipping for its memory and processing advantage compared to traditional flat clipping, regardless of the size of the model. Two group-wise clipping techniques were considered: per-layer

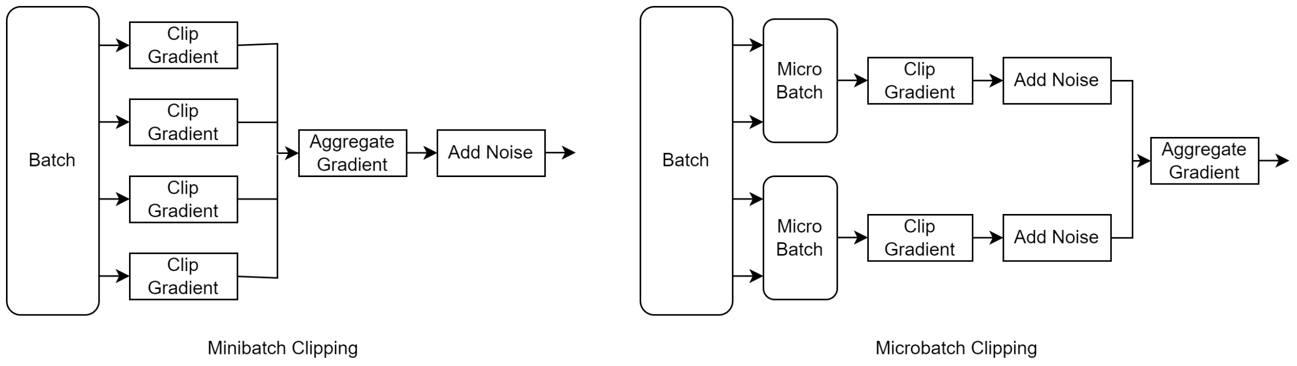


Figure 6: Minibatch Clipping vs Microbatch Clipping

clipping and per-device clipping.

4.2.1 Per-layer Clipping

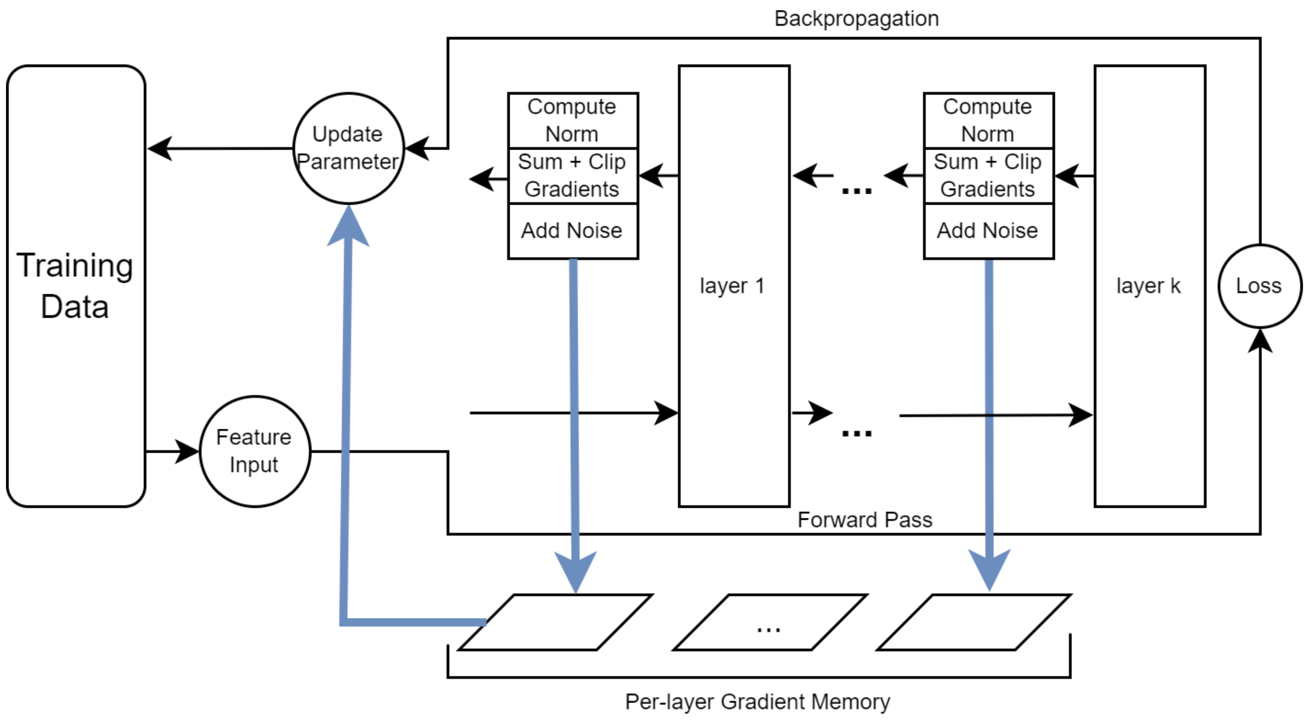


Figure 7: Per-layer Clipping

In per-layer clipping, per-example gradients of each layer in a neural network are clipped separately. Parameters of a neural network are grouped together and each of the K layers in the network is prescribed its own clipping threshold, C_k. This brings the main advantage of per-layer clipping, in which gradient clipping for a particular layer can be done as soon as the backpropagation returns the output gradient of that layer, allowing clipping to be done together with backpropagation, as opposed to flat clipping which requires backpropagation to completely finish first. Additionally, the process of summing and clipping of the per-example gradient can be combined as soon as the input activation, output gradients and per-example gradient norms are known. Figure 7 shows the entire training cycle and how each layer performs its gradient computation, as well as how each layer's gradients are stored separately to perform the parameter updates corresponding to each individual layer. Besides the computational advantage, this method does not require instantiating the per-example gradients in memory, and instead can be computed as needed. This technique is referred to as ghost clipping [39].

However, even though a computational advantage is noticed for per-layer clipping, utilizing a fixed clipping threshold for each layer ends up making the model perform worse compared to the traditional flat clipping method in certain scenarios. Observations during training reveal significant fluctuations in the magnitudes of per-layer gradients throughout the process. While gradients may initially be uniformly low, they tend to increase notably for layers closer to the input as training progresses. It is suggested that employing a fixed layer-wise clipping threshold eliminates the structural relationship between gradients of different layers, introducing an additional source of bias on top of the inherent bias associated with flat clipping techniques. As a solution to the performance problems of fixed per-layer clipping, adaptive clipping thresholds can alleviate the structural bias that comes with clipping gradients of different layers separately. In this case, the adaptive clipping technique focuses a quantile estimation, as discussed in Section 3.

4.2.2 Per-device Clipping

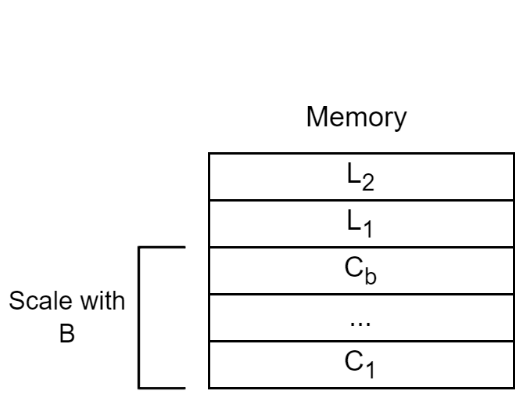
Another group-wise clipping technique explored in [6] is per-device clipping. The main motivation for this method is to make use of larger/better pre-trained model as past works have shown using them improves the privacy-utility trade off. The problem is that size of the model makes it so that the model weights cannot be fit in a single device (GPU), making it a difficult computational problem. The solution to this problem revolves around pipeline parallelism popularly used in non-private training [40]. The model is first partitioned into consecutive blocks/layers which is then assigned to its own accelerator (i.e., GPU). Forward computation is performed on micro-batches (split mini-batches) that chain together the local computations done on each model partition by communicating activation outputs between the GPUs. Backpropagation reverses this process but on each GPU, and intermediate forward computations are performed to reduce peak memory usage. Furthermore, parallel computation is enabled across GPUs to reduce overall training time. Once all micro-batches finish their forward and backward computation, SGD performs its parameter updates, and the GPUs are synchronized.

Since flat clipping requires computing per-example gradient norms in order to get the scaling factors for computing gradients, it becomes necessary to have additional communication between devices to get the per-example norms of their local gradients. This ends up adding extra overhead to the pipeline parallelism process [6]. There are three potential approaches to reduce communication, however both of them result in non-trivial slowdowns and increased complexity in the implementation. The first approach is to synchronize all the devices after the full backward pass is finished for each micro-batch. This allows each device to have the same gradient norms for computing the clipping scaling factor, but it ends up requiring as many synchronization steps as the amount of micro-batches in a mini-batch, which further reduces efficiency when micro-batches are large. The second option is to offload the unclipped local per-example gradients to the CPU, and to transport them back during synchronization. The problem here is the slow CPU-GPU data transfer rates ends up being costly. The last option is to re-materialize the local per-example gradient during synchronization, which ends up being costly as it requires a second backpropagation step to be performed.

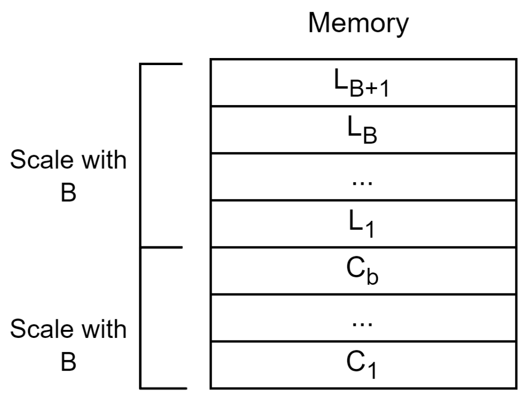
4.3 Quantifying Memory Requirements

Knowing the different ways clipping can be performed, it is important to quantify how much memory is required by each technique. Yousefpour et. al. [34] estimated the memory requirements in a per-example setting that uses the minibatching technique as follows:

$$\mathrm{M}_{\text{non-DP}} = bC + 2L$$



(a) Per-layer Clipping



(b) Mini-batch Clipping

Figure 8: Per-layer Clipping Memory vs Mini-batch Clipping Memory

$$\mathrm{M}_{\mathrm{DP}} = bC + (1+b)L$$

where L is the number of trainable parameters with each one being of size 1, C is the size of the features, label, and model output for a single data point, and M is the total memory usage for one forward and one backward pass on a batch of size b. The labels and outputs for b data points are expected to occupy memory of size bC and the model itself occupies memory of size L. In a non-private setting, the gradients are expected to occupy an additional memory of size L, meanwhile, for a private setting, the gradients are expected to occupy a memory of size bL due to the need to store per-sample gradients. If the batch size is greater than or equal to 1, it's possible to get an estimate of how much memory private training requires compared to non-private as a ratio of the number of trainable parameters and the size of the model input/outputs. This is given as:

$$\frac{\mathrm{M}_{\mathrm{DP}}}{\mathrm{M}_{\text{non-DP}}} = \begin{cases} \frac{bC+(1+b)L}{bC} = 1 + \frac{L}{C}, & \text{if } L/C << b \\ \frac{2+b}{3} = \frac{b}{3}, & \text{if } L/C = b \\ \frac{1+b}{2} = \frac{b}{2}, & \text{if } L/C >> b \end{cases}$$

He et. al. [6] compared the memory requirements between the per-layer clipping and the approach in [34] and found that per-layer clipping occupies a similar amount of memory as the non-private setting. This result is expected, given that the ghost clipping technique used in per-layer clipping eliminates the need to store per-sample gradients, making the memory requirement of storing the gradient independent of the batch size. This means that the traditional mini batching technique would require double the memory size compared to per-layer clipping, as traditional mini batching scales with the batch size twice, as opposed to only once with per-layer clipping. Figure 8 shows a comparison between the scaling factor with the batch size in terms of memory for per-layer clipping and mini-batch clipping. As an example, assume L and C both uses a memory unit of size 1, and training uses a batch size of 100. Using the equation above, we can estimate the memory required for traditional mini batch clipping with per-sample gradients to be 201. Meanwhile, for per-layer clipping, we would need a memory block of size 102 only. The difference will keep growing bigger with larger batch size, especially when we consider private training of large-scale models that require the use of multiple GPUs.

5 Individualized Privacy Budget

The majority of DP-SGD techniques assume a fixed privacy requirement (i.e., budget ε) for all data contributors, and focus on various adaptations of learning rate or budget consumption throughout the learning process to improve the privacy-accuracy trade-off (see Section 3). The privacy budget used in the training must adhere to the most stringent privacy requirement among all data contributors. However, in practice, various data contributors may have differing privacy expectations [41]. Data points from contributors with lower privacy requirements could potentially offer more valuable information for training machine learning models. Consequently, setting a uniform privacy budget across all data points may unnecessarily lower the training accuracy.

Recently [42], a novel research direction has surfaced, which focuses on supporting the enforcement of heterogeneous privacy constraints across the training dataset. Data points are partitioned into several groups, each with its own privacy budget. Essentially, the concept involves allocating greater privacy budgets to less sensitive data points and lower budgets to more sensitive ones. In this section, we review the several approaches for individualized privacy budget assignment.

The idea of individualized privacy constraints has been considered in the differential privacy literature even before DP-SGD emerged, in the context of statistical queries. The early work by Jorgensen et. al. [43] proposed two directions. The first one involves a two-step process: non-uniform sampling based on each tuple's specific privacy requirements, followed by applying a differentially private mechanism to the sampled dataset. This method effectively combines randomness from both steps to achieve personalized privacy guarantees. The second approach, inspired by the exponential mechanism developed by McSherry and Talwar [44], offers a more direct route to achieving Personalized Differential Privacy (PDP).

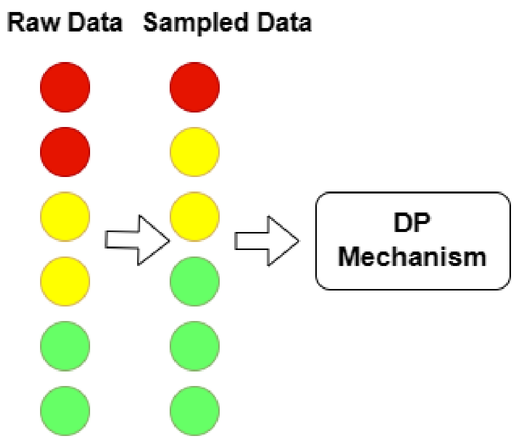
Later on, the work done by Li et. al. [45] explored two partitioning methods aimed at achieving PDP while optimizing individuals' privacy budgets and enhancing utility. These methods, termed privacy-aware and utility-based partitioning, group records with diverse privacy budgets into different bins. Each bin undergoes DP aggregate computation using its minimum privacy budget, and the perturbed results are aggregated in the final output.

A recent study [46] took an adaptive approach to PDP. The proposed adaptive framework for personalized differential privacy (AdaPDP) dynamically selects noise generation algorithms and determines parameters such as sensitivity and noise magnitude based on query functions, data distributions, and privacy settings, in order to maximize data utility. Additionally, AdaPDP conducts multiple rounds of utility-aware sampling to meet diverse privacy requirements for individual users.

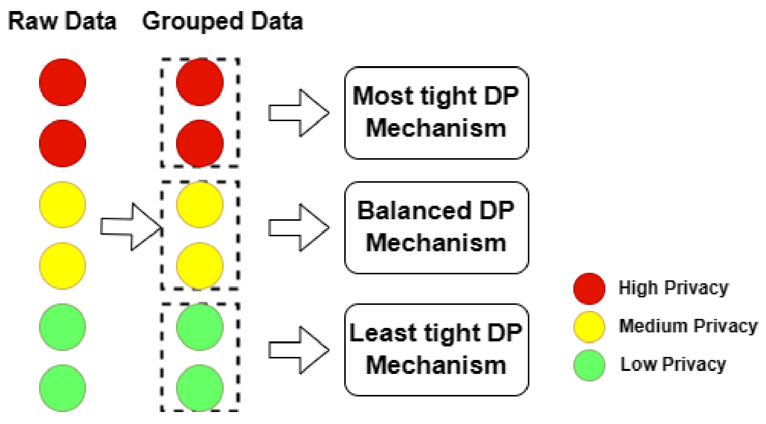
Most studies emphasize two key directions to achieve individualized privacy: sampling and grouping. In the former approach, sampling probabilities are assigned to data points inversely proportional to their privacy concerns, ensuring that records with higher privacy demand are sampled less frequently. In the latter, points with equal privacy concerns are clustered together, with higher-privacy groups undergoing more noise addition than lower concern groups. Figure 9 illustrates the two approaches.

The **upsampling** approach is a technique introduced in [42] in conjunction with the Private Aggregation of Teacher Ensembles (PATE) algorithm. PATE is an alternative to DP-SGD for training machine learning models on sensitive data while preserving individual privacy. It involves training multiple teacher models on disjoint subsets of the data, aggregating their predictions with differential privacy, and then training a student model using the noisy aggregated labels. PATE allows for effective model training while ensuring privacy protection, making it suitable for various applications in fields like healthcare and finance [8].

The upsampling mechanism relies on duplicating sensitive data such that overlapping data-subsets can be allocated to different teachers. Thereby, data with higher privacy budgets are used to train a higher number of teachers, thus revealing more information from data points with less restrictive privacy needs, and restricting the level of information derived from points with higher demands [42]. The algorithm ensures that points are duplicated by an integer according to the privacy budget ratios. Algorithm 1 shows how the upsampling factor is calculated for each data point.



(a) Sampling



(b) Grouping

Figure 9: Sampling Approach vs Grouping Approach

Algorithm 1: Upsampling method in PATE [42]

```
Input: Privacy budgets $\{\varepsilon_d\}$ for each data point $d$, precision $p \in \mathbb{N}$
Output: Upsampling factor $u_d$ for each data point $d$
$\{\varepsilon_1, \ldots, \varepsilon_j\} \leftarrow$ unique($\{\varepsilon_d\}$) ;          // Get unique budgets
for each $\varepsilon_j$ do
    $\bar{\varepsilon}_j \leftarrow \varepsilon_j \cdot 10^p$ ;                               // Upscale budgets
end
$D \leftarrow$ Greatest Common Divisor($\bar{\varepsilon}_1, \ldots, \bar{\varepsilon}_G$);
for each $\bar{\varepsilon}_j$ do
    $u_d \leftarrow \frac{\bar{\varepsilon}_j}{D}$;
end
```

Using the same upsampling technique, the work in [9] proposed a method that relies on sampling data points with different sample rates $q_1, \ldots, q_P$ depending on their individual privacy budgets. In this case, the noise multiplier σ_{sample} is fixed. Data points with higher privacy budgets (weaker privacy requirements) are assigned higher sampling rates than those with lower privacy budgets. This modifies the Poisson sampling process for DP-SGD to sample data points with higher privacy budgets within more training iterations. Using the minimum privacy budget of ϵ_1, the algorithm begins by initializing the σ_{sample} with σ from conventional DP-SGD, which is the noise multiplier needed for the privacy group G_1, which has the strongest privacy requirement of all groups. This is equivalent to instantiating σ_{sample} using the overall privacy groups' upper bound noise. Next, they employ a $getSampleRate$ function to determine the sample rates for the specified privacy parameters. The algorithm then reduces σ_{sample} repeatedly by a scaling factor that is marginally smaller than 1, and recalculates $q_1, \ldots, q_P$ until their weighted average approaches q (the traditional unified sampling rate).

The **grouping** approach is another technique used for individualized privacy budget assignment, which clusters together training examples with similar privacy requirements. In PATE, this step takes the form of a weighting mechanism [42] which adjusts how the aggregation of teacher votes is performed. It does this by assigning higher or lower weights to individual teachers' votes based on the privacy requirements of their training data points. Consequently, data points with similar privacy budgets ε_j, referred to as a privacy group g_j, must be assigned to the same teacher(s). Algorithm 2 outlines the process of assigning weights w_i to the teachers.

Algorithm 2: Weighting mechanism in [42]

```
Input: Privacy budget $\varepsilon_j$ and number of teachers $n_j$ for each privacy group j, $j \in \{1, \dots, G\}$, and total number of teachers $k$
Output: Weight $w_i$ for each teacher $t_i$
$E \leftarrow \sum_{j=1}^{G} \varepsilon_j$;
for each privacy group $g_j$ do
    $\bar{\varepsilon}_j \leftarrow \frac{\varepsilon_j}{E}$ ;                 // Relative privacy budget
    $\bar{n}_j \leftarrow \frac{n_j}{k}$ ;                   // Relative group size
    $\bar{w}_j \leftarrow \bar{\varepsilon}_j \cdot \bar{n}_j$ ;
end
$W \leftarrow \sum_{j=1}^{G} \bar{w}_j$;
for each privacy group $g_j$ do
    $w_j \leftarrow \frac{\bar{w}_j}{W} \cdot k$ ;               // Make sum of weights match $k$
    for each teacher $t_i$ with data from j do
        $w_i \leftarrow w_j$;
    end
end
```

Based on grouping, the work in [9] proposes a **scaling** method that adjusts the noise added to each gradient based on the privacy constraint of each data point. Current implementations of DP-SGD typically add noise to the sum of per-example clipped gradients over an entire mini-batch, resulting in the same amount of noise being added to all gradients. Instead, the authors of [9] introduce individualized clipping bounds $c_1, ..., c_P$ for each privacy level, effectively adjusting the scale of noise added on a per-example basis by modifying the sensitivity (clipping bound) of each example with a multiplier. Data points with higher privacy budgets (weaker privacy requirements) receive lower noise and higher clipping norms. As indicated in Equation (1) below, the clipping bound c does not directly impact the obtained ε; instead, the individualized privacy in the **scaling** approach results from the individual noise multipliers $\sigma_1, ..., \sigma_P$. The privacy guarantee ε depends on noise multiplier σ, sample rate q, the number of training iterations I, and the RDP order α [9]. This translates into utility gains thanks to the overall increase in the signal-to-noise ratio during training.

$$\varepsilon \leq I \cdot 2q^2 \frac{\alpha}{\sigma^2} \tag{1}$$

However, directly implementing individual noise multipliers per privacy group in **scaling** degrades training performance, because noise is added per mini-batch, while sampling and gradient clipping are performed per data point in DP-SGD. Restricting mini-batches to contain only data points from the same privacy group, which share the same noise multiplier, would lead to a loss of gains in the privacy-utility trade-offs resulting from subsampling. Therefore, while relying on mini-batches that contain data points with different privacy requirements (i.e., different noise multipliers), one can specify a single fixed noise multiplier σ scale.

To overcome this limitation, the work in [9] does not set noise multipliers $\sigma_1, ..., \sigma_P$ directly, but instead obtains them indirectly through individualized clipping bounds $c_1, ..., c_P$. In conventional DP-SGD, a gradient clipped to c obtains noise with standard deviation $\sigma * c$. In the **scaling** approach, gradients are clipped to $c_p = s_p * c$ with a per-privacy group scaling factor s_p, obtaining noise multiplier $\sigma_p c_p$. Noise is added according to $\sigma_{scale} c$ to all mini-batches. Thus, the effective noise scale σ_p of each data point becomes $\sigma_p = \frac{1}{s_p} * \sigma_{scale}$. Data points with higher privacy budgets have $s_p > 1$, receiving lower noise multipliers, and vice- versa for lower privacy budgets.

Assuming all users in the training dataset have equal privacy concerns, certain training examples contribute more significantly to the learning process than others. This discrepancy means that some examples may pose higher privacy risks than others. A recent study by [47] introduces a novel approach to privacy assurance termed output-specific individual differential privacy. This method is designed to analyze the privacy guarantees of individual data points within models trained using DP-SGD. Investigations from [47] reveal that many data points benefit from stronger privacy guarantees than initially anticipated under the worst-case scenario, and that there is a robust correlation between a data point's privacy level and its associated training loss.

The fundamental concept behind output-specific (ϵ, δ)-DP involves defining the privacy parameter ϵ as a function of both the outputs and the specific target data point. In essence, for a given data point d and a subset of outcomes $A \subset O$, an algorithm $A : D \to O$ satisfies output-specific individual $(\epsilon(A, d), \delta)$-DP for d at A if certain predefined conditions are met. To operationalize this approach, an algorithm is introduced to compute per-step RDP for each example using estimated individual gradient norms. Additionally, this algorithm facilitates the updating of individual gradient norms and the accumulated RDP. Furthermore, to streamline the computational cost associated with individual privacy assessment, two parameters are introduced: the frequency K for computing batch gradient norms and the decision of whether to round individual gradient norms using a small constant r.

6 Experiments

We conducted a brief benchmarking of the various studied techniques for adaptive and individualized DP-SGD, the purpose of which is two-fold: first, some of the experimental runs in the original papers introducing these techniques were conducted under different datasets and/or different parameter settings, making it difficult to directly compare approaches head-to-head. Second, we wanted to validate the results presented by the authors in their papers, and thus conduct a reproducibility study. For all experiments, we use the RDP moments accountant as defined by Abadi et. al. in [2], based on the concept of Renyi Differential Privacy discussed in Section 2.

We used two prominent datasets in our runs:

- MNIST: introduced by LeCun et al. [48] in 1998, consists of handwritten digit images. Specifically, it contains 60,000 training examples and 10,000 test examples, where each example is a 28×28 grayscale image.
- CIFAR-10: Developed by Krizhevsky and Hinton in 2009 [49], it consists of color images classified into ten distinct classes, including objects like airplanes, cars, and birds. It includes 50,000 training images and 10,000 test images, with each image being a 32×32 pixel RGB image.

6.1 Adaptive DP-SGD Results

First, we investigated three prominent techniques for noise magnitude decaying discussed in Section 3.1.1: Time Decay, Exponential Decay and Polynomial Decay [20]. These experiments were performed using the same parameters settings as in the original work from 3.1.1 ($\sigma_{initial} = 10$ for all decaying strategies) on the MNIST dataset. Figure 10 and Table 2 summarize the results. Our experiments show that these techniques assign very low privacy budget (ϵ) to the earlier iterations with a controlled increase over time, allowing the training gradients to be injected with higher noise at the start of training, and reducing the noise in later iterations, as training converges. Figure 10 illustrates the privacy budget consumption rate for all mentioned techniques along with standard DP-SGD for $\sigma_{fixed} = 8$. Among all approaches, Time Decay consumes the most privacy-budget, and at the highest rate. Table 2 presents the testing accuracy achieved using the above-mentioned decaying strategies. Time Decay achieves the highest accuracy of 91.50% after consuming approximately $\epsilon = 1.0$ while Polynomial Decay and Exponential Decay achieved 90.66% and 89.00% respectively, for the same aggregate privacy budget consumption.

Next, we experimented with adapting learning rate (η) and clipping threshold (C), according to the methodology described in [7] and [3] (described in Sections 3.2.2 and 3.3, respectively). Noise magnitude was set at $\sigma = 2.0$. Figure 11 shows the test accuracy of each technique when training on the MNIST dataset. Among the

Table 2: Accuracy Comparison of Noise Decay Strategies

Stopping Criterion	Stopping Threshold	Polynomial Decay	Time Decay	Exponential Decay
Epochs	100	91.67%	92.18%	87.41%
ϵ	1	90.66%	91.50%	89.00%

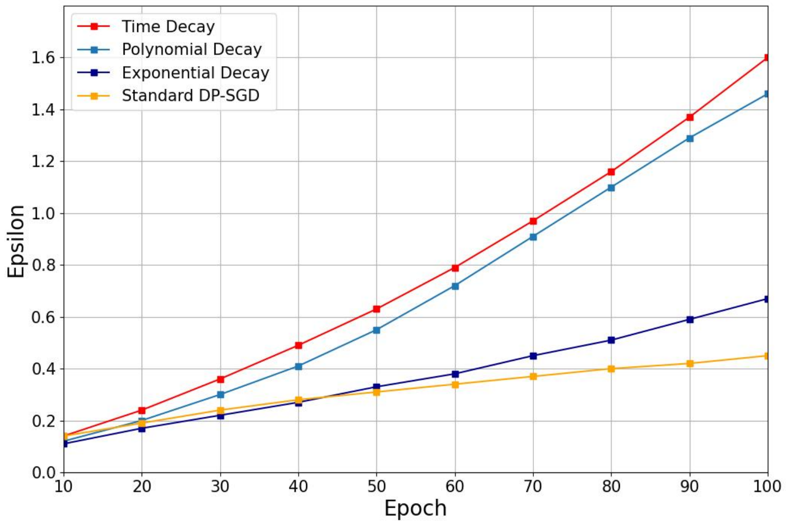


Figure 10: Adaptive Privacy Budget Consumption Rate

three, adaptive learning rate produced the best accuracy (91.90%) after consuming privacy budget $\epsilon = 1.0$, while adaptive clipping achieved 85.37% accuracy with overall privacy budget consumption of $\epsilon = 0.83$ – a tighter privacy guarantee than standard DP-SGD which achieved accuracy of 86.00% after consuming $\epsilon = 1.0$.

For comparison purpose, we tested the performance of the noise magnitude time decay methodology discussed earlier using the same parameters that have been used for the latest adaptation experiments. The noise magnitude time decay seems to be the most promising among all, as it yielded an accuracy of 92.05% with $\epsilon = 1.0$ in a shorter training time.

We conducted the same experiment on the CIFAR-10 dataset, using the same architecture and parameters used in [7], and $\sigma = 1.2$ for all techniques except for the exponential noise decaying approach, for which we used $\sigma_{initial} = 2.0$, in order to prevent very high privacy budget consumption. Figure 12 summarizes the results. Experiments on CIFAR-10 confirm the superiority of learning rate adaptation, which obtained the highest accuracy of 62.52% with $\epsilon = 2.0$ going up to 64.97% with $\epsilon = 3.0$. Clipping threshold adaptation produced slightly better improvements with an average of 7% additional accuracy when compared to standard DP-SGD at similar privacy levels. One important observation from this set of experiments is the performance of noise magnitude decay. Specifically, exponential noise decay is able to achieve only marginally better accuracy compared with the standard DP-SGD under similar budget consumption.

6.2 Individualized Privacy Budget Results

We evaluate two different individualized privacy (IDP) approaches proposed by Boenisch et. al. in [9], namely grouping and sampling. We divide the training dataset into three partitions according to the privacy requirements

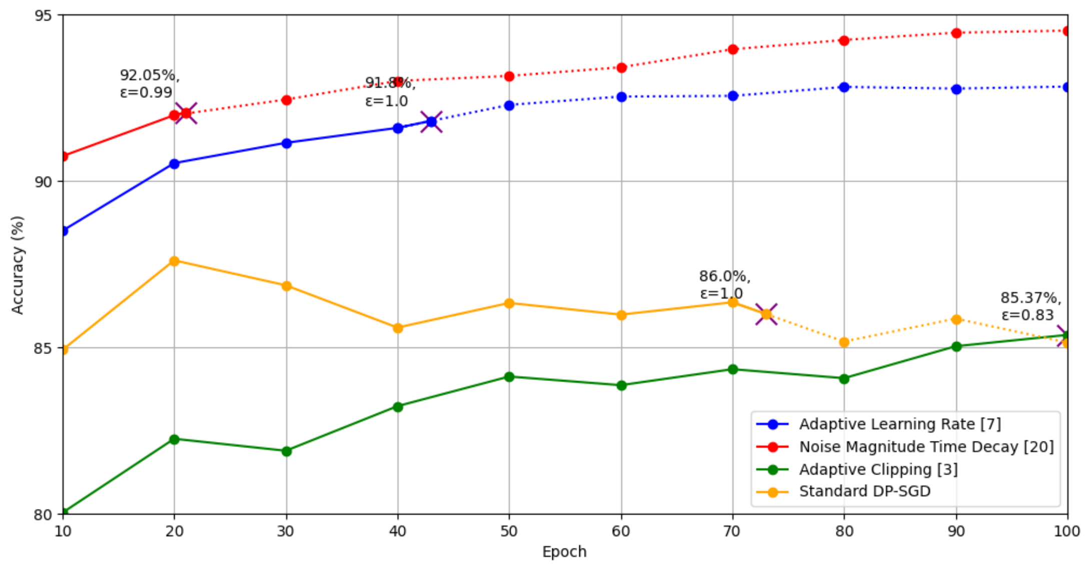


Figure 11: Accuracy of Adaptive Techniques on MNIST

illustrated in Table 3. The two distributions correspond to the settings used in [9] and [50], respectively.

First we present results obtained on the MNIST dataset. According to [9], the sampling approach uses individual sampling rates = {0.005, 0.009, 0.013} for the three different privacy level groups respectively, with $\sigma_{sample} = 1.368$. In the scaling approach, the corresponding individual clipping thresholds are {0.141, 0.236, 0.299} with $\sigma_{scale} = 1.545$. After training the model using both approaches, scaling obtained accuracy of 97.63% while sampling achieved 96.79% accuracy, both outperforming previously mentioned DP-SGD related approach. The results match the ones presented in the original IDP work from [9].

We also investigated results on CIFAR-10, which were not reported previously. We include two more extreme cases of privacy requirements, cases 2 and 3 in Table 3. Case 1 is the only one used in the original work from [9]. The second case has a more pronounced variability in privacy concerns, with one group having very tight privacy requirements, while the others are more loose. Finally, the third case has two groups with relatively tight privacy requirements, and a third with almost no privacy concerns.

Figures 13 and 14 show the privacy consumption rates for each group in case 1. As shown in Figure 15 and Table 4, both grouping and sampling approaches were able to deal with the extreme cases, with the scaling approach giving better performance in cases 2 and 3. The most extreme cases lead to slowest convergence in training. This is a result of overfitting, as the sampled examples originate mostly in group 3, and thus the model finds it difficult to generalize results to less seen examples in groups 1 and 2.

7 Conclusions

In this article, we reviewed several categories of prominent DP-SGD techniques with respect to several criteria, such as strategies for adaptive hyperparameter tuning, data management considerations, and individualized privacy requirements. With the current advent of machine learning in virtually all application domains, DP-SGD is expected to become increasingly deployed in practice. Therefore, it is important to understand well its privacy-accuracy trade-off, its underlying influential factors, and how to adapt private learning to obtain a good compromise between protection, accuracy and performance. In future work, we plan to extend our review to

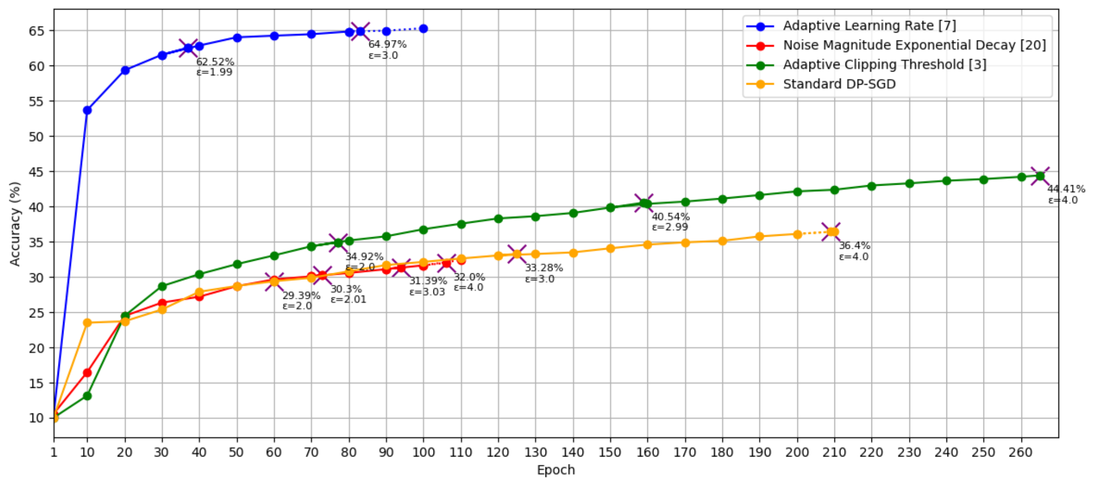


Figure 12: Accuracy of Adaptive Techniques on CIFAR-10

Table 3: Individualized Data Distributions

Group #	**Percentage of Data**	**Case 1**	**Case 2**	**Case 3**
Group 1 (High Privacy)	34%	$\epsilon = 1.0$	$\epsilon = 1.0$	$\epsilon = 1.0$
Group 2 (Medium Privacy)	43%	$\epsilon = 2.0$	$\epsilon = 10.0$	$\epsilon = 2.0$
Group 3 (Low Privacy)	23%	$\epsilon = 3.0$	$\epsilon = 20.0$	$\epsilon = 20.0$

Table 4: Performance of IDP-SGD on CIFAR-10

Approach	**Case 1**	**Case 2**	**Case 3**
Sampling	59.03%	59.49%	59.7%
Scaling	58.26%	67.64%	63.07%

techniques for private learning in large language models, which present a different set of challenges, due to the multiple ways in which unstructured language from individuals can carry sensitive information into the final model.

References

[1] C. Dwork, “Differential privacy,” in Automata, Languages and Programming (M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, eds.), (Berlin, Heidelberg), pp. 1–12, Springer Berlin Heidelberg, 2006.

[2] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS ’16, (New York, NY, USA), p. 308–318, Association for Computing Machinery, 2016.

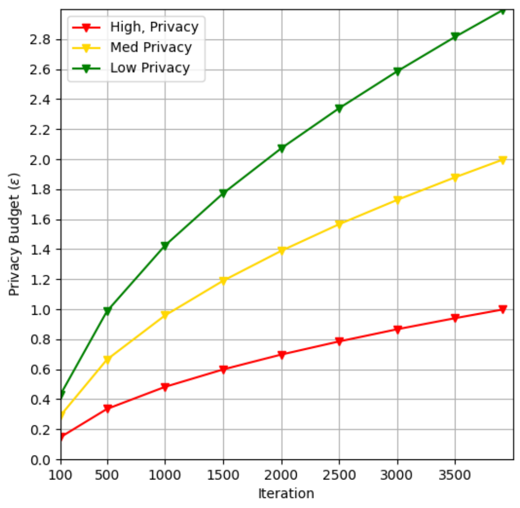


Figure 13: Individualized privacy budget consumption over time (Sampling)

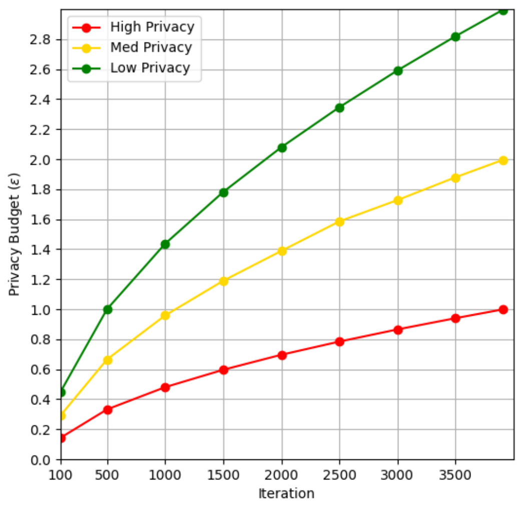


Figure 14: Individualized privacy budget consumption over time (Scaling)

[3] G. Andrew, O. Thakkar, B. McMahan, and S. Ramaswamy, "Differentially private learning with adaptive clipping," in Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual (M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, eds.), pp. 17455–17466, 2021.

[4] C. Chen and J. Lee, "Stochastic adaptive line search for differentially private optimization," in 2020 IEEE International Conference on Big Data (Big Data), pp. 1011–1020, IEEE, 2020.

[5] V. Pichapati, A. T. Suresh, F. X. Yu, S. J. Reddi, and S. Kumar, "AdaCliP: Adaptive clipping for private SGD," CoRR, vol. abs/1908.07643, 2019.

[6] J. He, X. Li, D. Yu, H. Zhang, J. Kulkarni, Y. T. Lee, A. Backurs, N. Yu, and J. Bian, "Exploring the limits of differentially private deep learning with group-wise clipping," in The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023, OpenReview.net, 2023.

[7] A. Koskela and A. Honkela, "Learning rate adaptation for differentially private learning," in Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (S. Chiappa and R. Calandra, eds.), vol. 108 of Proceedings of Machine Learning Research, pp. 2465–2475, PMLR, 26–28 Aug 2020.

[8] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson, "Scalable private learning with PATE," ICLR, vol. 1050, p. 24, 2018.

[9] F. Boenisch, C. Mühl, A. Dziedzic, R. Rinberg, and N. Papernot, "Have it your way: Individualized privacy assignment for DP-SGD," Advances in Neural Information Processing Systems, vol. 36, 2024.

[10] T. Dalenius, "Towards a methodology for statistical disclosure control," 1977.

[11] Y. Zhao and J. Chen, "A survey on differential privacy for unstructured data content," vol. 54, no. 10s, 2022.

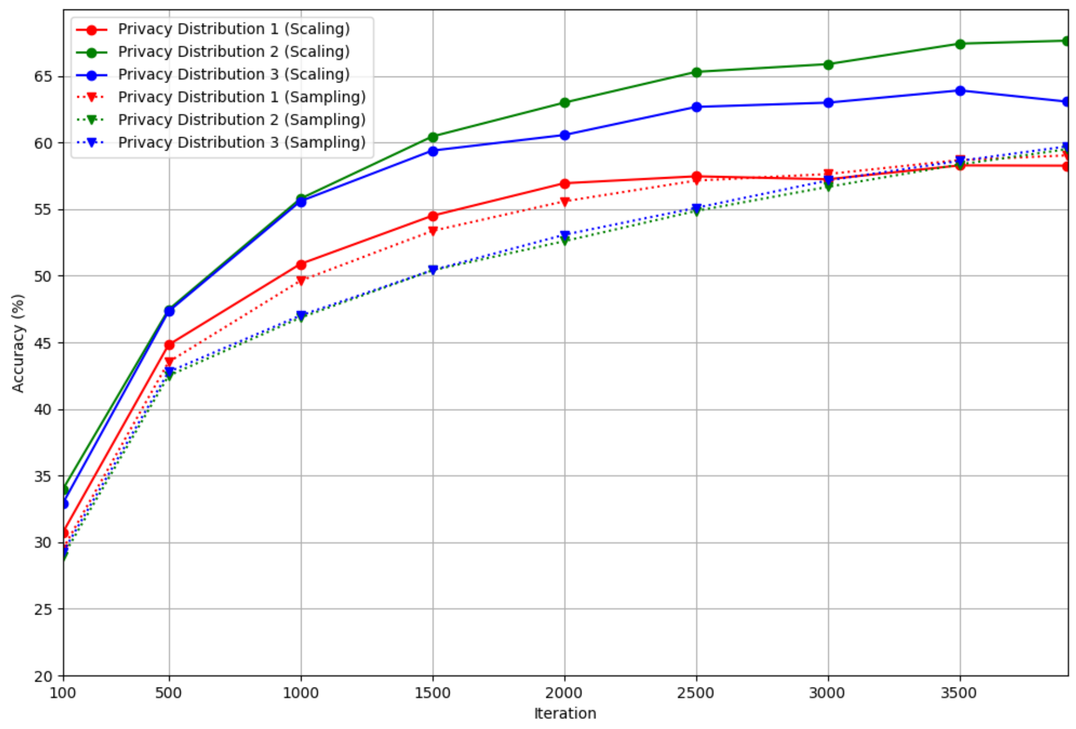


Figure 15: Testing accuracy for IDP-SGD techniques on CIFAR-10

[12] C. Dwork, "Differential privacy: A survey of results," in Theory and Applications of Models of Computation (M. Agrawal, D. Du, Z. Duan, and A. Li, eds.), (Berlin, Heidelberg), pp. 1–19, Springer Berlin Heidelberg, 2008.

[13] J. Lee and D. Kifer, "Concentrated differentially private gradient descent with adaptive per-iteration privacy budget," in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1656–1665, 2018.

[14] I. Mironov, "Rényi differential privacy," in 2017 IEEE 30th computer security foundations symposium (CSF), pp. 263–275, IEEE, 2017.

[15] T. van Erven and P. Harremos, "Rényi divergence and kullback-leibler divergence," IEEE Transactions on Information Theory, vol. 60, no. 7, pp. 3797–3820, 2014.

[16] R. M. Gower, N. Loizou, X. Qian, A. Sailanbayev, E. Shulgin, and P. Richtárik, "SGD: General analysis and improved rates," in International conference on machine learning, pp. 5200–5209, 2019.

[17] Y. Tian, Y. Zhang, and H. Zhang, "Recent advances in stochastic gradient descent in deep learning," Mathematics, vol. 11, no. 3, p. 682, 2023.

[18] M. Gong, Y. Xie, K. Pan, K. Feng, and A. K. Qin, "A survey on differentially private machine learning," IEEE computational intelligence magazine, vol. 15, no. 2, pp. 49–64, 2020.

[19] D. Fay, S. Magnússon, J. Sjölund, and M. Johansson, "Adaptive hyperparameter selection for differentially private gradient descent," Transactions on Machine Learning Research, 2023.

[20] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, "Differentially private model publishing for deep learning," in 2019 IEEE Symposium on Security and Privacy (SP), pp. 332–349, 2019.

[21] J. Liu and K. Talwar, "Private selection from private candidates," in Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, (New York, NY, USA), p. 298–309, Association for Computing Machinery, 2019.

[22] A. Roth and T. Roughgarden, "Interactive privacy via the median mechanism," in Proceedings of the Forty-Second ACM Symposium on Theory of Computing, STOC '10, (New York, NY, USA), p. 765–774, Association for Computing Machinery, 2010.

[23] M. Hardt and G. N. Rothblum, "A multiplicative weights mechanism for privacy-preserving data analysis," in 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, pp. 61–70, 2010.

[24] J. Hong, Z. Wang, and J. Zhou, "Dynamic privacy budget allocation improves data efficiency of differentially private gradient descent," in Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, FAccT '22, (New York, NY, USA), p. 11–35, Association for Computing Machinery, 2022.

[25] B. Polyak, "Gradient methods for the minimisation of functionals," USSR Computational Mathematics and Mathematical Physics, vol. 3, no. 4, pp. 864–878, 1963.

[26] K. L. van der Veen, R. Seggers, P. Bloem, and G. Patrini, "Three tools for practical differential privacy," 2018.

[27] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh, "Large batch optimization for deep learning: Training BERT in 76 minutes," in International Conference on Learning Representations, 2020.

[28] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "ZeRO: memory optimizations toward training trillion parameter models," in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '20, IEEE Press, 2020.

[29] G. Pleiss, D. Chen, G. Huang, T. Li, L. van der Maaten, and K. Q. Weinberger, "Memory-efficient implementation of DenseNets," CoRR, vol. abs/1707.06990, 2017.

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, PyTorch: an imperative style, high-performance deep learning library. Red Hook, NY, USA: Curran Associates Inc., 2019.

[31] C. Dupuy, R. Arava, R. Gupta, and A. Rumshisky, "An efficient DP-SGD mechanism for large scale NLP models," CoRR, vol. abs/2107.14586, 2021.

[32] K. Lv, Y. Yang, T. Liu, Q. Gao, Q. Guo, and X. Qiu, "Full parameter fine-tuning for large language models with limited resources," 2024.

[33] J. Lee and D. Kifer, "Scaling up differentially private deep learning with fast per-example gradient clipping," Proc. Priv. Enhancing Technol., vol. 2021, no. 1, pp. 128–144, 2021.

[34] A. Yousefpour, I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao, G. Cormode, and I. Mironov, "Opacus: User-friendly differential privacy library in pytorch," CoRR, vol. abs/2109.12298, 2021.

[35] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," CoRR, vol. abs/1609.04836, 2016.

[36] N. Ponomareva, S. Vassilvitskii, Z. Xu, B. McMahan, A. Kurakin, and C. Zhang, "How to DP-fy ML: A practical tutorial to machine learning with differential privacy," KDD '23, (New York, NY, USA), p. 5823–5824, Association for Computing Machinery, 2023.

[37] M. Patel and E. J. Yuen, "Farewell, CUDA OOM: Automatic gradient accumulation." `https://www.databricks.com/blog/farewell-oom`.

[38] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, OpenReview.net, 2018.

[39] X. Li, F. Tramèr, P. Liang, and T. Hashimoto, "Large language models can be strong differentially private learners," in The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022, OpenReview.net, 2022.

[40] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. X. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, and Z. Chen, "GPipe: Efficient training of giant neural networks using pipeline parallelism," in Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada (H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, eds.), pp. 103–112, 2019.

[41] R. Cummings, G. Kaptchuk, and E. M. Redmiles, "" i need a better description": An investigation into user expectations for differential privacy," in Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pp. 3037–3052, 2021.

[42] F. Boenisch, C. Mühl, R. Rinberg, J. Ihrig, and A. Dziedzic, "Individualized PATE: Differentially private machine learning with individual privacy guarantees," Proceedings on Privacy Enhancing Technologies, 2023.

[43] Z. Jorgensen, T. Yu, and G. Cormode, "Conservative or liberal? personalized differential privacy," in Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 1203–1217, ACM, 2015.

[44] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07), pp. 94–103, IEEE, 2007.

[45] H. Li, L. Xiong, Z. Ji, and X. Jiang, "Partitioning-based mechanisms under personalized differential privacy," in Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), vol. 10234, p. 615, 2017.

[46] B. Niu, Y. Chen, B. Wang, Z. Wang, and F. Li, "AdaPDP: Adaptive personalized differential privacy," in IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, pp. 1–10, IEEE, 2021.

[47] D. Yu, G. Kamath, J. Kulkarni, T.-Y. Liu, J. Yin, and H. Zhang, "Individual privacy accounting for differentially private stochastic gradient descent," Transactions on Machine Learning Research, 2023.

[48] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[49] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, “Do cifar-10 classifiers generalize to cifar-10?,” arXiv preprint arXiv:1806.00451, 2018.

[50] M. Alaggan, S. Gambs, and A.-M. Kermarrec, “Heterogeneous differential privacy,” Journal of Privacy and Confidentiality, vol. 7, no. 2, 2016.

Does Differential Privacy Impact Bias in Pretrained Language Models?

Md. Khairul Islam[1], Andrew Wang[1], Tianhao Wang[1], Yangfeng Ji[1],
Judy Fox[1], Jieyu Zhao[2]
[1] University of Virginia, [2] University of Southern California
{mi3se, ajw7uhj, tianhao, yj3fs, cwk9mp}@virginia.edu, jieyuz@usc.edu

Abstract

Differential privacy (DP) is applied when fine-tuning pre-trained language models (LMs) to limit leakage of training examples. While most DP research has focused on improving a model's privacy-utility tradeoff, some find that DP can be unfair to or biased against underrepresented groups. In this work, we extensively analyze the impact of DP on bias in LMs. We find differentially private training can increase the model bias against protected groups w.r.t AUC-based bias metrics. DP makes it more difficult for the model to differentiate between the positive and negative examples from the protected groups and other groups in the rest of the population. Our results also show that the impact of DP on bias is affected by both the privacy protection level and the underlying distribution of the dataset.

1 Introduction

In a data-driven world, an appropriate dataset is critical for fair and responsible decision-making and analysis. However, inequalities in the world or limitations in the collection process may restrict the database coverage across different minority groups and representations [26]. Identifying these discriminations [28] and insufficient coverages [29] helps reduce the database bias which can impact the models trained on them. Social media and online conversation platforms are places where millions of user relies on such text databases everyday. The user's confidentiality and fair decisions are very crucial for these natural language processing (NLP) tasks.

Pretrained transformer-based language models such as BERT [1] have led to significant advancements in research. Much of the success of natural language models (LMs) ultimately derives from the vast data used to train these models. However, the use of a large training dataset raises concerns about data privacy, where the model can be used to detect the presence of sensitive information in the training data. To defend against these attacks, Differentially private (DP) training techniques [2, 3] have been used during the model training or fine-tuning process [4]. These techniques ensure that a model does not leak sensitive training data. Otherwise, an attacker can extract the dataset [5] using inference attacks.

However, recent works in data privacy indicate that DP training may cause machine learning models to become more biased [6, 7, 8]. However, most of these works focus on the computer vision domain or tabular datasets. With the wide usage of NLP models and the urgency to realize trustworthy NLP, we need to understand

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

whether we can obtain an NLP model equipped with both privacy and fairness, especially for the pre-trained language models. An NLP model is considered biased when the model is unable to perform on protected social groups equally as well as on others. For example, prior research has demonstrated a coreference resolution model can behave very differently for different demographic groups [9, 10]. DP may introduce bias because it steers a model away from relying on a select few data points, causing that model to attend poorly to social groups that are underrepresented in the training data.

In this work, we explore the impact of differential privacy on model bias in the pre-trained BERT language model. The degree of which can be tuned by adjusting the privacy budget parameter. We train the model with different privacy budgets and measure the bias across six identity subgroups using multiple metrics. We consider bias in the context of the toxic language detection task, which has been shown to produce biased models [11]. We choose two popular datasets, Jigsaw Unintended Bias [12] and the Measuring Hate Speech from UCBerkeley [13]. We use both prediction and probability-based bias metrics to analyze the effect of DP on the bias from different perspectives. We then investigate them in each identity group for any discriminatory behavior against them.

Contributions: In this work, we present a detailed analysis of the impact of DP training on bias in fine-tuned language models. We present our results on two popular hate speech datasets by training our models at different privacy levels and analyzing how it affects the model bias. We show that DP training makes the model more biased in AUC-based metrics. DP also negatively affects the model's utility when adopted to pertained language models. Our findings will give new insights into the privacy and bias trade-off, which can help NLP researchers incorporate DP into their works. The code for our work can be found at: https://github.com/khairulislam/DP-on-NLP-Bias.

2 Related Work

Fairness is critical to data-driven systems [26] to make more accurate decisions at a larger scale while managing millions of user data. Misrepresentations of minorities and insufficient data coverage across different modalities [29] introduce bias in modern database systems. [26] proposed data-centric approaches to identify and resolve these issues in data. [28] formalized the discriminations as a database repair problem and provided sufficient conditions to train fair classifiers. Their solution correctly captures subtle fairness violations in the data and provides provable fairness guarantees about classifiers trained on them. [29] proposed an efficient approach to identify regions with insufficient coverage across different groups over multiple relational tables in a database. The solution can efficiently identify inequalities in the collected data and help with a more fair solution.

Prior research has shown from a theoretical perspective that DP has a detrimental effect on model fairness [6, 14]. [6] assume the conditions of "pure DP" [2], and demonstrate that such a model cannot achieve perfect equal opportunity between social groups. [14] finds that model fairness has a disproportionately negative impact on accuracy for certain social groups.

In computer vision, recent works have empirically investigated the effects of DP on model fairness in models with more realistic privacy settings. Empirical analyses have found that DP can worsen accuracy for certain subgroups in image recognition tasks [7, 8] and synthetic data generation tasks [15]. [16] showed both DP-SGD and PATE have a disparate impact on the under-represented groups, but PATE has a significantly less disproportionate impact on utility compared to DP-SGD.

Bagdasaryan and Shmatikov [7] found that DP can worsen model bias in sentiment analysis. However, they only considered a single bias metric (accuracy degradation between privileged and unprivileged groups) on a single dataset using a glove-based model. In this work, we analyze pre-trained BERT models on multiple datasets using multiple bias metrics. Balancing between fairness and privacy can significantly impact using private models in practice.

Private-FairNR [6] algorithm approximately satisfies fairness for a private learner sampling hypothesis. [17] formally guaranteed the privacy of extracted text representation, while also helping model fairness. They aimed to protect the test phase privacy of end users while adopting local DP (LDP) with the Laplace mechanism.

3 Model Bias in NLP

Evaluating biases in NLP models requires a metric over some demographic groups. In this section, we describe the terminology for those groups and the metrics for bias evaluation.

3.1 Terminology

Protected attributes refer to sensitive attributes such as gender and race that should not be used to discriminate against individuals [18]. *Bias* occurs when a model experiences a degradation in performance when inferring examples pertaining to certain social groups implied by a protected attribute such as gender or race. In our calculations of bias, we refer to a *subgroup* as the social group whose bias we are measuring and *background* as the rest of the evaluation set [12]. *Prediction-based bias metrics* calculate the bias against the protected attributes using the model's predicted label (e.g. positive/negative), whereas *Probability-based bias metrics* use the prediction probability to calculate bias. These definitions of bias metrics are done following [19].

3.2 Protected Attributes

Bias in NLP has been well studied within the protected attributes of *gender* [20] and *race* [11]. Following this, we examined bias for sensitive attributes *gender* and *race*. In *gender* attribute the identity subgroups are *male/men*, *female/women*, and *transgender*. For *race* attribute the identity subgroups are *white*, *black*, and *asian*.

3.3 Bias Evaluation Metrics

A *degradation in performance* indicative of model bias can be measured in different ways. We consider metrics such as equality of odds metrics because of their prolific use in other NLP model fairness literature [18, 12, 21] and Bias-AUC because of its use as the benchmark in the Jigsaw Unintended Bias competition. We summarize all the different bias evaluation metrics we consider in Table 1. The implementations follow [18, 12] and [21]. More details about these metrics are in Appendix 6.

Bias Metric	**Formulation**	**Short form**
Demographic Parity [18]	$1 - \lvert p(\hat{Y}=1 \mid A=1) - p(\hat{Y}=1 \mid A=0)\rvert$	parity
Equality of Opportunity (w.r.t $Y=1$)	$1 - \lvert p(\hat{Y}=1 \mid Y=1, A=1) - p(\hat{Y}=1 \mid Y=1, A=0)\rvert$	EqOpp1
Equality of Opportunity (w.r.t $Y=0$)	$1 - \lvert p(\hat{Y}=1 \mid Y=0, A=1) - p(\hat{Y}=1 \mid Y=0, A=0)\rvert$	EqOpp0
Equality of Odds [18]	$0.5 \times [EqOpp0 + EqOpp1]$	EqOdd
Protected Accuracy [21]	$p(\hat{Y}=y \mid Y=y, A=1), y \in \{0,1\}$	p-acc
Subgroup AUC [12]	$AUC(D_g^- + D_g^+)$	
Background Pos, Subgroup Neg [12]	$AUC(D^+ + D_g^-)$	BPSN
Background Neg, Subgroup Pos [12]	$AUC(D^- + D_g^+)$	BNSP

Table 1: X, Y, A denotes the input, label, and sensitive attribute (e.g. male, female). $\hat{Y}$ and p are the model's prediction and the output probability. All metrics are in the range $[0-1]$ and a higher value is better (less bias). D_g^+ and D_g^- are the set of positive and negative examples in the identity subgroup g. D^+ and D^- are the set of positive and negative examples outside g.

4 Differential Privacy

Differential privacy (DP) [2] aims to preserve privacy using a quantifiable protection guarantee and acceptable utility in the context of statistical information disclosure. It is the *de facto* definition for privacy. In the context of our work, we use the notion of (ε, δ)-privacy. Following [2], if we have some arbitrary operation $\mathcal{A}$ with output space S and two datasets D, D' that differ in only a single record, then we can formulate (ε, δ)-privacy as $\Pr(\mathcal{A}(D) \in S) \leq e^{\epsilon} \Pr(\mathcal{A}(D') \in S) + \delta$.

By limiting any effect due to the inclusion of one individual's data (by the parameter ϵ), the DP notion approximates the effect of "opting-out": whether an individual's data is included or not does not influence the result much, thus the fact that the individual participated in the data release is protected.

To satisfy DP, noise is added to the aggregated-level results such that an individual's information disclosure is bounded. Our implementation in this paper uses the Gaussian mechanism [22] to guarantee (ϵ, δ)-DP.

DP in machine learning: When training models with DP, perturbations are added to the gradients (i.e., clipping the gradients and then adding Gaussian noise) [3]. More specifically, during the t-th iteration the optimizer will compute noisy gradients as:

$$g^t = \frac{1}{|B|}(\sum_{x_i \in B} \hat{g}_i^t + \mathcal{N}\left(0, \sigma^2 C^2 I\right)),$$

where B is a subsampled batch used to compute the gradients, w^{t-1} is the current model before t-th iteration, σ is noise multiplier,

$$\hat{g}_i^t = \nabla f(x_i; w^{t-1}) \min\{1, \frac{C}{\|\nabla f(x_i; w^{t-1})\|_2}\}$$

(i.e., each gradient is clipped by C, so that $\sum \hat{g}_i^t$ has bounded ℓ_2-sensitivity and we can use the Gaussian mechanism to ensure DP), and g^t is the (noisy) gradient used to update the model.

Training a model requires multiple training epochs. Our formulation of DP is amenable to this practice. If we have k operations that satisfy some ε privacy constraint, we can combine those operations and maintain DP for $O(\sqrt{k}\epsilon)$. So we refer to ϵ as the *privacy budget* of a privacy-preserving algorithm.

Impact of DP Methods on Fairness: Gaussian Mechanism introduces enough noise so that the contribution of individual data points to model decision-making is limited. However, a byproduct of this approach is that the distinguishing features of underrepresented social groups within the dataset can be "smoothed over." Thus, we conjecture that the DP model attends disproportionately worse to the underrepresented social groups and is thus biased. In what follows, we present evidence supporting the fact that DP negatively impacts the model fairness.

5 Datasets

We choose two popular toxicity detection datasets for our study, Jigsaw Unintended Bias [12] and UCBerkeley Hate Speech [13]. Both datasets (1) have target labels so that we can use supervised learning, (2) are for text classification using NLP techniques, and (3) have annotated social groups for all examples.

5.1 Jigsaw Unintended Bias

The Jigsaw Unintended Bias dataset was developed to learn and minimize any unintended bias against different identities that a machine learning model learns to predict toxicity [27] from online comments. Here toxicity is defined as anything rude or disrespectful that can make someone leave a discussion. The dataset collected by Jigsaw and Google has annotations for demographic groups by disability, gender, race or ethnicity, religion, and

Group	Jigsaw				UCBerkeley			
	Train		Test		Train		Test	
	class 0	class 1	class 0	class 1	class 0	class 1	class 0	class 1
Male	3187	3375	1792	320	2361	796	502	171
Female	3950	3639	2252	350	4852	2305	1042	511
Transgender	158	287	103	26	882	244	196	51
White	1507	3612	825	353	1694	643	378	132
Black	901	2369	515	246	2103	1568	483	337
Asian	358	282	196	21	831	207	195	53
Total	144334	72167	89543	7777	19376	7618	4142	1643

Table 2: Distribution of identities in both datasets. Total is the class distribution in the dataset after pre-processing. Class 1 is for toxic and 0 for non-toxic.

sexual orientation. The complete dataset has about 2 million examples. We report the label distribution for each identity in Table 2.

Train/Validation/Test split: We undersampled the training dataset using a 2:1 ratio between the non-toxic and toxic labels. Due to computing resource limitations, we halved the training set, preserving the 2:1 label distribution. This yielded a training set with 144,334 non-toxic examples and 72,167 toxic examples. We use the pre-existing splits from the original source for the validation and test data. This yielded a test and a validation set each with 97,320 examples.

5.2 UCBerkeley Hate Speech

This dataset [1] is a collection of online comments from three major social media platforms (YouTube, Twitter, and Reddit), labeled by human annotators through crowd-sourcing [13]. It provides a unique way to measure hate speech at eight theorized qualitative from genocidal hate speech to counter speech. The dataset comes with annotations for the targeted group in the comment text.

Pre-processing: The original dataset has 135,556 comments and the annotations for 'hatespeech' contain 3 classes: 0 for neutral or counter speech, 1 when the annotator is unclear, and 2 for hate speech. For the simplicity of the experiment, we dropped comments with label 1, converting the task to a binary classification where hate speech is a positive class and non-hate speech is negative. The dataset also had multiple annotations per comment. We aggregated the annotations for each comment. If any comment had the same label at least from 50% of the annotators, then it was chosen as true, otherwise false. After aggregation, we had 38,564 comments left. Additionally, the dataset contains transgender identity labels split into multiple groups (transgender_men, transgender_women, transgender_unspecified). We combined them together in a single transgender column for bias calculation.

Train/Validation/Test split: We randomly split the aggregated data into train, validation, and test sets using a 70:15:15 ratio.

[1] https://huggingface.co/datasets/ucberkeley-dlab/measuring-hate-speech

6 Experimental Setup

Model: We use the pre-trained BERT-base-uncased model from HuggingFace [2] to perform all our experiments in this section. For training the model on downstream tasks we choose only to train the last three layers (final encoder layer, pooler, classifier). The rest of the layers were frozen, yielding 7.6 M trainable parameters out of a total of 109M. We choose to train only these layers because: 1) DP is more effective when applied to fewer layers, and 2) we can utilize BERT's rich pre-trained embeddings.

Input texts were tokenized using the BERT-base-uncased tokenizer from HuggingFace. The comment texts were generally not very lengthy, so we kept the maximum sequence length to 128 across both datasets. The batch size was set to 64.

Optimization: We use the Adam optimizer with cross-entropy loss and learning rate 10^{-3}. We train each model for a maximum of 10 epochs. At each epoch, the trained model is evaluated on the validation set and saved if the F1 score improves. Early stopping patience was 3. We also used a learning rate scheduler (ReduceLROnPlateau) to reduce the learning rate by a factor of 0.1 if the validation F1 score does not improve for more than one epoch.

Privacy: We use the Pytorch Opacus library [23]. It provides a privacy engine to train models with DP-SGD [3]. DP-SGD was chosen since it is the most widely used one in the related works [7, 14, 24], supports iterative training process and is available as a framework. We use the *make_private_with_epsilon* method offered by the library, which takes as input the model to be trained, optimizer, training data, number of epochs, target ϵ, target δ and maximum gradient norm. The target epsilon is the privacy budget we want to achieve. For a reasonable privacy guarantee, ϵ should be set below 10 [3] and this setting has been followed in other applications of DP on NLP [7, 24, 17]. For our task, we experimented with five different target epsilons 0.5, 1.0, 3.0, 6.0, and 9.0. The smaller the value the more private the model is. This will show us the change in model behavior at different privacy levels.

Evaluation: We tune the training process using the F1 score on the validation set, then checkpoint the best model based on that, and finally use that model to evaluate the test set. We have presented the final test results in Section 7. Each experiment is run three times with arbitrarily chosen random seeds 2022, 42, and 888. The average score is reported in Table 3.

Bias Evaluation Metrics We use the following metrics for calculating bias during our experiments:

- *Equality of Odds (EqOdd) [18]:* Widely used to measure unequal treatments against protected groups in the dataset. The metric combines the disparity in false positive and true positive rates for two social groups in the same protected class.
- *Demographic Parity (Parity) [18]:* Enforces the model's prediction to be independent of the protected attribute. The metric computes the difference in likelihood between unprotected or protected examples to be classified as positive.
- *Subgroup AUC, BPSN, and BNSP [12]:* The Subgroup AUC, BPSN, and BNSP metrics measure the unintended bias in the dataset based on the AUC metric. AUC is threshold agnostic, unlike equality of odds or other prediction-based metrics that require converting model predictions into positive or negative classes using some threshold. The choice of threshold can change the results and provide misleading measurements. These metrics can be used to find new and potentially subtle biases in models.

[2]https://huggingface.co/bert-base-uncased

7 Results

7.1 Overall Results

Here we present the impact of adding DP on the overall model utility for both datasets. Table 3 shows that the model utility decreases with stricter privacy (smaller ϵ). However, for the UCBerkeley dataset, the false positive rate increases, and the recall drops significantly. This shows the model predicts fewer positive cases with added privacy. The recall drop is also significant for Jigsaw. This decrease in overall performance also impacts the performance of the identity subgroups.

Metric	**Jigsaw - Privacy Budget (ϵ)**						**UCBerkeley - Privacy Budget (ϵ)**					
	∞	≤ 9.0	≤ 6.0	≤ 3.0	≤ 1.0	≤ 0.5	∞	≤ 9.0	≤ 6.0	≤ 3.0	≤ 1.0	≤ 0.5
Acc	0.911	0.887	0.886	0.884	0.871	0.870	0.807	0.787	0.787	0.785	0.779	0.772
F1	0.593	0.522	0.518	0.508	0.459	0.440	0.647	0.554	0.559	0.539	0.523	0.480
AUC	0.946	0.920	0.918	0.913	0.886	0.872	0.855	0.813	0.819	0.814	0.802	0.790
FPR	0.080	0.102	0.103	0.105	0.113	0.111	0.120	0.086	0.089	0.079	0.082	0.069
TPR	0.809	0.768	0.763	0.751	0.686	0.642	0.623	0.469	0.476	0.443	0.427	0.371

Table 3: Overall model performance. The results are best for the non-DP training ($\epsilon \rightarrow \infty$) and worst at the most strict privacy budget, $\epsilon \leq 0.5$.

7.2 Prediction Based Metrics

Equality of Odds, parity, and protected accuracy are prediction-based bias metrics. They calculate the bias score based on the model's prediction. We present the results in Table 4. For each identity, we report the best and the worst results, and the privacy budget that achieves that result. The closer these scores are to 1, the less the bias is.

		Jigsaw			**UCBerkeley**		
Group		**EqOdd**	**parity**	**p-acc**	**EqOdd**	**parity**	**p-acc**
Male	min	0.894 (6.0)	0.852 (1.0)	0.741 (1.0)	0.955 (∞)	0.763 (∞)	0.765 (1.0)
	max	0.928 (0.5)	0.872 (∞)	0.801 (∞)	0.983 (0.5)	0.868 (0.5)	0.799 (∞)
Female	min	0.932 (9.0)	0.851 (1.0)	0.785 (9.0)	0.937 (∞)	0.890 (∞)	0.717 (0.5)
	max	0.940 (0.5)	0.872 (∞)	0.822 (∞)	0.957 (3.0)	0.929 (0.5)	0.756 (∞)
Transgender	min	0.818 (9.0)	0.842 (1.0)	0.674 (∞)	0.910 (∞)	0.740 (∞)	0.815 (9.0)
	max	0.952 (0.5)	0.863 (∞)	0.785 (0.5)	0.962 (0.5)	0.848 (0.5)	0.839 (∞)
White	min	0.734 (9.0)	0.853 (1.0)	0.588 (6.0)	0.917 (9.0)	0.752 (∞)	0.769 (9.0)
	max	0.842 (0.5)	0.875 (∞)	0.647 (0.5)	0.940 (1.0)	0.851 (0.5)	0.800 (∞)
Black	min	0.777 (∞)	0.847 (1.0)	0.636 (9.0)	0.812 (3.0)	0.836 (∞)	0.761 (0.5)
	max	0.901 (0.5)	0.871 (∞)	0.697 (0.5)	0.855 (∞)	0.924 (0.5)	0.821 (∞)
Asian	min	0.916 (∞)	0.842 (1.0)	0.814 (9.0)	0.871 (∞)	0.737 (∞)	0.823 (0.5)
	max	0.976 (0.5)	0.863 (∞)	0.859 (∞)	0.894 (0.5)	0.844 (0.5)	0.847 (∞)
Trend $\epsilon \downarrow$		$\uparrow$	$\downarrow$	$\uparrow\downarrow$	$\uparrow$	$\uparrow$	$\downarrow$

Table 4: Prediction Based Bias (Jigsaw). The privacy budget (ϵ) for each metric is mentioned in the parentheses. **The trends are not monotonic and can be mixed.** Smaller ϵ means stricter privacy.

Table 4 shows several trends depending on the dataset and metric. The equality of odds always improves with a strict privacy budget (small ϵ). However, this is due to a significant drop in recall (Figure 2) for most groups. They are reduced to a smaller score range. Thus the TPR difference becomes smaller, improving the EqOpp1.

The trend in demographic parity is the opposite in both datasets. With a stricter privacy budget, parity decreased in the Jigsaw (2-3%) but increased in the UCBerkeley dataset (4-11%). An increase in this value indicates that the model's decision of whether the comment is toxic or not, is more independent of the protected group [18]. We show in Section 8.4 that DP increases positive predictions in Jigsaw and decreases them in UCBerkeley. More positive predictions increase the probability of disparity among different subgroups of Jigsaw. Similarly in UCBerkeley dataset, since there are fewer positive predictions from the model, the disparity based on positive outcomes decreases too.

The protected accuracy has mixed trends in the Jigsaw dataset, changing in either direction. In the UCBerkeley dataset, there is a 2-5% drop with DP training. The detailed plots for these metrics at each privacy budget and for each identity are available in our GitHub repo.

7.3 Probability Based Metrics

This section presents the bias calculated using the metrics presented by [12]. These metrics are dependent on the model's prediction probability, hence better representing the bias in the model's confidence. They are also threshold agnostic, unlike prediction-based metrics.

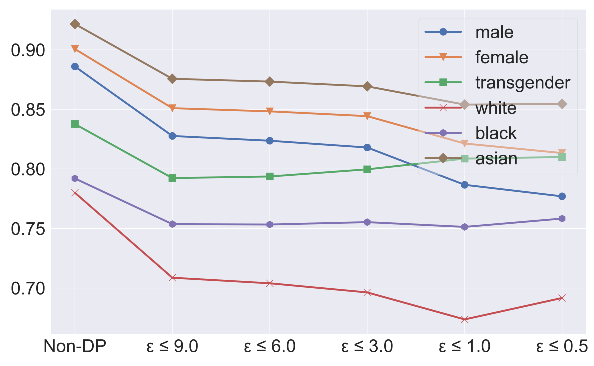
Figure 1 shows that for stricter privacy (smaller ϵ), both BNSP and BPSN drop significantly for most identities. A drop in BNSP means the scores for positive examples in these subgroups are lower than the scores for other negative examples in the background data. These examples would likely appear as false negatives within the subgroup at many thresholds [12].

Similarly, a drop in BPSN means scores for negative examples in these subgroups are higher than scores for other positive examples in the background. These examples would likely appear as false positives within these subgroups at many thresholds [12]. A decrease in the subgroup AUC score shows that the model can not understand and separate the positive and negative examples within the subgroup. These drops between non-DP training and training with DP at $\epsilon \leq 0.5$ are highlighted in Table 5, showing an increase in bias at stricter privacy budgets, compared to non-DP training.

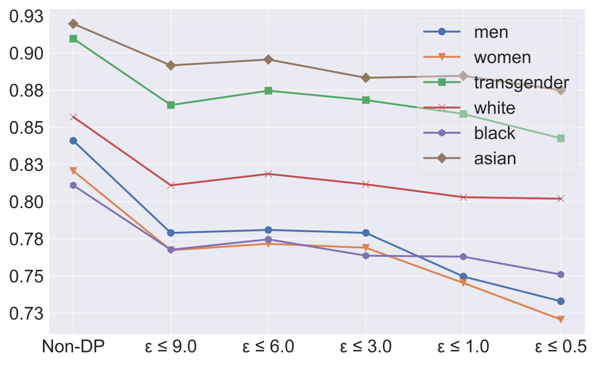
	Jigsaw			UCBerkeley		
Group	**Δ Subgroup AUC**	**Δ BPSN**	**Δ BNSP**	**Δ Subgroup AUC**	**Δ BPSN**	**Δ BNSP**
Male	**0.097**	0.064	**0.109**	**0.081**	0.036	**0.108**
Female	0.079	0.067	0.087	0.067	**0.037**	0.100
Transgender	0.008	**0.082**	0.028	0.033	0.017	0.067
White	0.063	0.058	**0.088**	0.057	0.070	0.055
Black	**0.081**	**0.098**	0.034	0.036	0.047	**0.060**
Asian	0.016	0.036	0.067	**0.069**	**0.086**	0.045

Table 5: Decrease in probability-based bias from non-DP training to training with $\epsilon \leq 0.5$. The biggest drop along each metric column for each sensitive attribute (race, gender) is in bold. The DP model is 4-11% more biased in several identity groups.

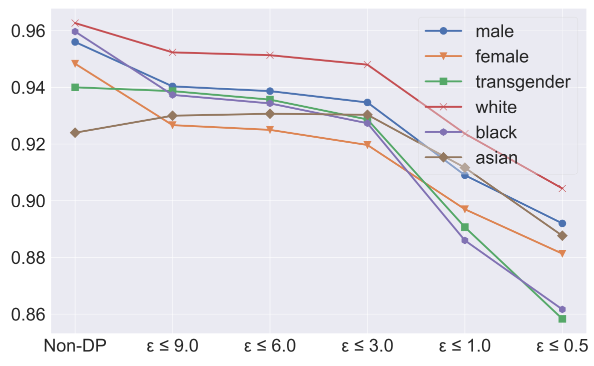
Figure 1 shows some interesting cases. In the Jigsaw dataset, white and black identities have much lower AUC and BPSN scores compared to others. Similarly in the UCBerkeley dataset, men and women have much lower AUC and BPSN scores than other identities. This shows that the DP models more often tend to label non-toxic comments mentioning these identities as toxic, compared to the non-DP models. Additionally, DP amplifies the difference in the AUC gap between white and Asian subgroups in Jigsaw and white and black subgroups in UCBerkeley. The non-DP model already had a gap in AUC between them, but DP increases it.



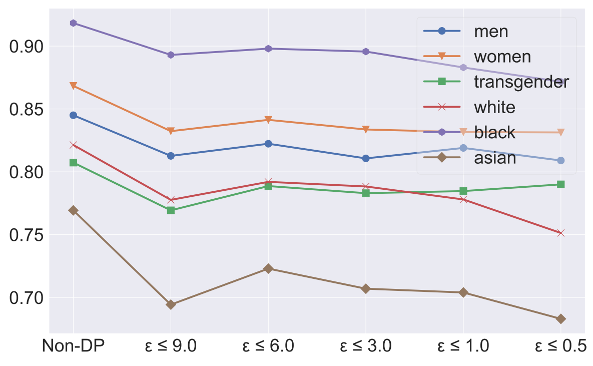
(a) BPSN (Jigsaw)



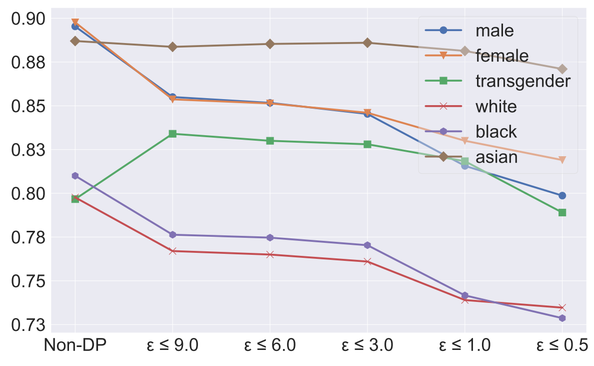
(b) BPSN (UCBerkeley)



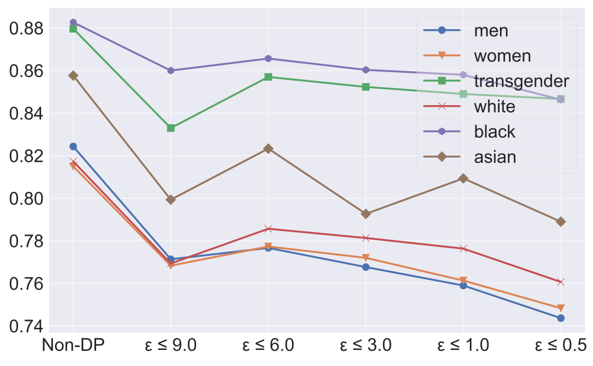
(c) BNSP (Jigsaw)



(d) BNSP (UCBerkeley)



(e) Subgroup AUC (Jigsaw)



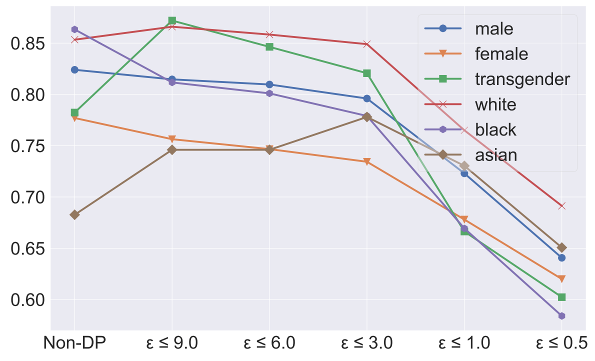
(f) Subgroup AUC (UCBerkeley)

Figure 1: AUC based bias [12]. BNSP for Jigsaw and BPSN for UCBerkeley have a significant drop in value with a much smaller ϵ. The larger the drop, the more biased the model w.r.t that metric.

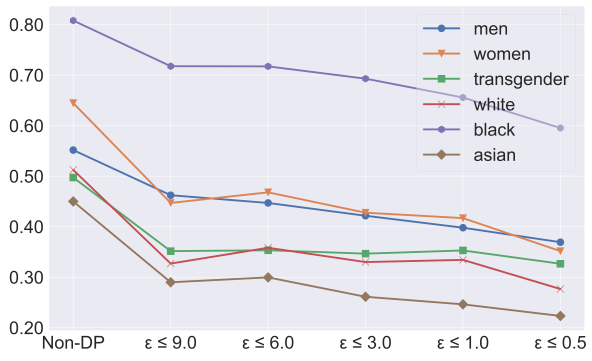
8 Discussion

8.1 DP's Positive Impact on Equality of Odds and Opportunities.

Equality of odds is a function of relative true positive and false positive rates between a subgroup and the background population. As such we investigate why the addition of noise does not decrease relative TPR (recall) and FPR. DP adds noise in the training phase, adversely affecting overall model performance (Table 3). The model experiences a degradation in the recall for all social groups as the privacy setting increases. So recall values grow more similar. This minimizes the difference in TPR between a subgroup and a background population, contributing to an overall improvement in equality of odds. However, such a trend does not necessarily indicate a decrease in bias but instead indicates that a model is losing its ability to differentiate between the positive and negative classes. Figure 2 shows that the recall drops significantly for private training.



(a) Jigsaw - Recall



(b) UCBerkeley - Recall

Figure 2: Recall for each subgroup at different privacy budgets. Differential private training significantly reduces the model's ability to predict target toxic comments.

8.2 DP's Impact on Probability-based Bias.

The fall in overall model AUC scores also affects the subgroup AUC, BPSN, and BNSP, as shown in Table 5. The model makes more mistakes in differentiating the positive and negative examples between the subgroups and the background data, even within the subgroup itself. Thus introducing substantial bias against those subgroups at different prediction thresholds. [12] showed these subtle biases in the toxicity datasets might not be captured by prediction-based metrics like EqOdd, which depends on prediction thresholds. So we have prioritized the AUC-based bias metrics (subgroup AUC, BPSN, BNSP) over the other ones to investigate any potential bias.

8.3 Bias Gap between Groups

In this section, we show how much DP affects the gap between bias metrics of a pair of groups from the same attribute (race, gender). Table 6 shows the results in terms of AUC-based bias metrics for a non-private ($e\epsilon \to \infty$) and a private ($\epsilon \leq 0.5$)model. The results show that in many cases DP significantly widens the gap between bias metrics of different groups. For most other cases the gap changes slightly. And in rare occasions, there is a drop in the gap.

8.4 Predicted Label Distribution.

We found DP has opposite effects on the two datasets about total toxic comments being predicted, as shown in Table 7. In Jigsaw, increasing privacy in the training increases the number of toxic predictions. In the UCBerkeley dataset, the toxic predictions decrease with an increased privacy budget.

It can be attributed to how the dataset is distributed. [13] targeted an even distribution of labeled comments across different hate intensity levels, focused on finding more hate speech examples, whereas in Jigsaw there was no such filtering when creating the dataset. So the model trained on UC Berkeley is more skewed toward hate comments, whereas with Jigsaw it is the opposite. Adding DP introduces both noise and gradient clipping during the training and thus reduces this skewness. Finally, raises the plausibility of predicting opposite examples.

Subgroup		Jigsaw						UCBerkeley					
		Δ BPSN		Δ BNSP		ΔAUC		Δ BPSN		Δ BNSP		Δ AUC	
		$\epsilon \to \infty$	$\epsilon \leq 0.5$	$\epsilon \to \infty$	$\epsilon \leq 0.5$	$\epsilon \to \infty$	$\epsilon \leq 0.5$	$\epsilon \to \infty$	$\epsilon \leq 0.5$	$\epsilon \to \infty$	$\epsilon \leq 0.5$	$\epsilon \to \infty$	$\epsilon \leq 0.5$
Male	Female	0.015	0.036	0.008	0.011	0.003	0.020	0.020	0.012	0.023	0.022	0.009	0.004
Male	Trans.	0.048	0.033	0.016	0.034	**0.098**	0.010	0.069	**0.110**	0.380	0.190	0.056	**0.103**
Female	Trans.	**0.063**	0.003	0.008	0.023	**0.101**	0.030	0.089	**0.122**	0.061	0.041	0.065	**0.099**
White	Black	0.012	**0.066**	0.003	0.042	0.012	0.006	0.046	0.051	0.097	0.012	0.066	**0.085**
White	Asian	0.142	0.163	0.039	0.016	0.089	**0.136**	0.063	0.073	0.052	0.068	0.041	0.028
Black	Asian	0.130	0.097	0.036	0.026	0.077	**0.142**	0.109	0.129	0.149	**0.189**	0.025	0.057

Table 6: Difference in AUC-based bias metrics between groups of the same attribute (race, gender). Cases where the gap between bias changed significantly are in bold.

Budget	**Jigsaw**		**UCBerkeley**	
(ϵ)	True	False	True	False
∞	0.138	0.862	0.227	0.737
9.0	0.155	0.845	0.195	0.805
6.0	0.156	0.844	0.200	0.801
3.0	0.156	0.845	0.183	0.817
1.0	0.159	0.841	0.180	0.820
0.5	0.153	0.847	0.155	0.845
Trend $\epsilon \downarrow$	$\uparrow$	$\downarrow$	$\downarrow$	$\uparrow$

Table 7: Predicted Label Distribution

8.5 Limitations

We make our observations based on toxicity and hate speech detection tasks. However, bias in NLP has also been investigated in other tasks like coreference resolution [20], sentiment analysis [25], and question answering. Whether trends found in our results persist in those tasks too, is something to be explored for future works. We consider six diverse identity subgroups across two protected attributes (race, and gender). There exist more sensitive attributes in the dataset like religion and sexual orientation which are not explored here but can be explored in future works. We saw similar trends in bias across both selected attributes and the identity subgroups. Even with new attributes or subgroups, the trends should persist similarly.

9 Conclusion

In this work, we explore how differential privacy affects the bias in NLP models. We found DP increases model bias and the impact of that increase varies across different identities. We perform our empirical analysis on two hate/toxic language detection datasets. We evaluated the gender and racial bias of the model using different bias metrics for models trained at different privacy budgets (ϵ). We found that (Table 5) stronger privacy budgets cause the model to have more difficulty distinguishing between the positive/negative examples in the identity subgroup from negative/positive examples in other subgroups at different prediction thresholds [12]. We also observe an increase in equality of odds at a much stricter privacy level, mainly because the recall drops significantly for each group, reducing the difference between them. However, the protected accuracy also drops in most cases. Our overall observations confirm that DP increases bias in the NLP models for hate speech detection, and the researchers need to be aware of this bias when adding privacy to NLP models.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[2] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Theory of cryptography conference, pages 265–284. Springer, 2006.

[3] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pages 308–318, 2016.

[4] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, et al. Differentially private fine-tuning of language models. arXiv preprint arXiv:2110.06500, 2021.

[5] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In 30th USENIX Security Symposium (USENIX Security 21), pages 2633–2650, 2021.

[6] Rachel Cummings, Varun Gupta, Dhamma Kimpara, and Jamie Morgenstern. On the compatibility of privacy and fairness. In Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization, UMAP'19 Adjunct, page 309–315, New York, NY, USA, 2019. Association for Computing Machinery.

[7] Eugene Bagdasaryan and Vitaly Shmatikov. Differential privacy has disparate impact on model accuracy. CoRR, abs/1905.12101, 2019.

[8] Amartya Sanyal, Yaxi Hu, and Fanny Yang. How unfair is private learning ? 06 2022.

[9] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Gender bias in coreference resolution: Evaluation and debiasing methods. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 15–20, 2018.

[10] Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. Gender bias in coreference resolution. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 8–14, 2018.

[11] Thomas Davidson, Debasmita Bhattacharya, and Ingmar Weber. Racial bias in hate speech and abusive language detection datasets. In Proceedings of the Third Workshop on Abusive Language Online, pages 25–35, Florence, Italy, August 2019. Association for Computational Linguistics.

[12] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. In Companion proceedings of the 2019 world wide web conference, pages 491–500, 2019.

[13] Chris J Kennedy, Geoff Bacon, Alexander Sahn, and Claudia von Vacano. Constructing interval variables via faceted rasch measurement and multitask deep learning: a hate speech application. arXiv preprint arXiv:2009.10277, 2020.

[14] Cuong Tran, My H. Dinh, and Ferdinando Fioretto. Differentially private deep learning under the fairness lens. CoRR, abs/2106.02674, 2021.

[15] Georgi Ganev, Bristena Oprisanu, and Emiliano De Cristofaro. Robin hood and matthew effects–differential privacy has disparate impact on synthetic data. arXiv preprint arXiv:2109.11429, 2021.

[16] Archit Uniyal, Rakshit Naidu, Sasikanth Kotti, Sahib Singh, Patrik Joslin Kenfack, Fatemehsadat Mireshghallah, and Andrew Trask. Dp-sgd vs pate: Which has less disparate impact on model accuracy? arXiv preprint arXiv:2106.12576, 2021.

[17] Lingjuan Lyu, Xuanli He, and Yitong Li. Differentially private representation for NLP: Formal guarantee and an empirical study on privacy and fairness. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 2355–2365, 2020.

[18] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. Advances in neural information processing systems, 29, 2016.

[19] Paula Czarnowska, Yogarshi Vyas, and Kashif Shah. Quantifying Social Biases in NLP: A Generalization and Empirical Comparison of Extrinsic Fairness Metrics. Transactions of the Association for Computational Linguis-

tics,9:1249–1267, 11 2021.

[20] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Ryan Cotterell, Vicente Ordonez,and Kai-Wei Chang. Gender bias in contextualized word embeddings. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 629–634, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[21] Charan Reddy, Deepak Sharma, Soroush Mehri, Adriana Romero-Soriano, Samira Shabanian, and Sina Honari. Benchmarking bias mitigation algorithms in representation learning through fairness metrics. In Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1), 2021.

[22] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science, 9(34):211–407, 2014.

[23] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, et al. Opacus: User friendly differential privacy library in pytorch. arXiv preprint arXiv:2109.12298, 2021.

[24] Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. Large-scale differentially private bert. arXiv preprint arXiv:2108.01624, 2021.

[25] Svetlana Kiritchenko and Saif Mohammad. Examining gender and race bias in two hundred sentiment analysis systems. In Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics, pages 43–53, 2018.

[26] Shahbazi, Nima, Mahdi Erfanian, and Abolfazl Asudeh. "Coverage-based Data-centric Approaches for Responsible and Trustworthy AI." Data Engineering (2024): 3.

[27] cjadams, Daniel Borkan, inversion, Jeffrey Sorensen, Lucas Dixon, Lucy Vasserman, nithum. (2019). Jigsaw Unintended Bias in Toxicity Classification. Kaggle. https://kaggle.com/competitions/jigsaw-unintended-bias-in-toxicity-classification.

[28] Salimi, Babak, Bill Howe, and Dan Suciu. "Database repair meets algorithmic fairness." ACM SIGMOD Record 49.1 (2020): 34-41.

[29] Lin, Yin, Yifan Guan, Abolfazl Asudeh, and H. V. Jagadish. "Identifying insufficient data coverage in databases with multiple relations." Proceedings of the VLDB Endowment 13, no. 11 (2020)..

Data Engineering

It's FREE to join!

TCDE

tab.computer.org/tcde/

The Technical Committee on Data Engineering (TCDE) of the IEEE Computer Society is concerned with the role of data in the design, development, management and utilization of information systems.

- Data Management Systems and Modern Hardware/Software Platforms
- Data Models, Data Integration, Semantics and Data Quality
- Spatial, Temporal, Graph, Scientific, Statistical and Multimedia Databases
- Data Mining, Data Warehousing, and OLAP
- Big Data, Streams and Clouds
- Information Management, Distribution, Mobility, and the WWW
- Data Security, Privacy and Trust
- Performance, Experiments, and Analysis of Data Systems

The TCDE sponsors the International Conference on Data Engineering (ICDE). It publishes a quarterly newsletter, the Data Engineering Bulletin. If you are a member of the IEEE Computer Society, you may join the TCDE and receive copies of the Data Engineering Bulletin without cost. There are approximately 1000 members of the TCDE.

Join TCDE via Online or Fax

ONLINE: Follow the instructions on this page:

www.computer.org/portal/web/tandc/joinatc

FAX: Complete your details and fax this form to **+61-7-3365 3248**

Name ____________________

IEEE Member # ____________________

Mailing Address ____________________

____

Country ____________________

Email ____________________

Phone ____________________

TCDE Mailing List

TCDE will occasionally email announcements, and other opportunities available for members. This mailing list will be used only for this purpose.

Membership Questions?

Xiaoyong Du
Key Laboratory of Data Engineering and Knowledge Engineering
Renmin University of China
Beijing 100872, China
duyong@ruc.edu.cn

TCDE Chair

Xiaofang Zhou
School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane, QLD 4072, Australia
zxf@uq.edu.au