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

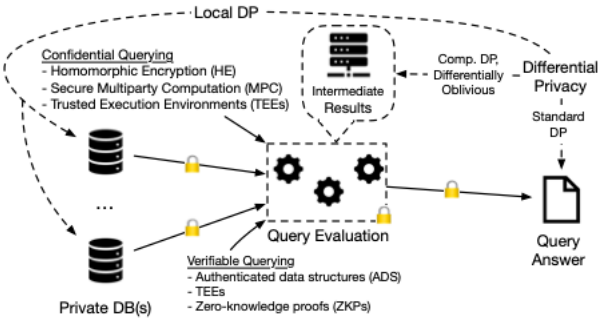
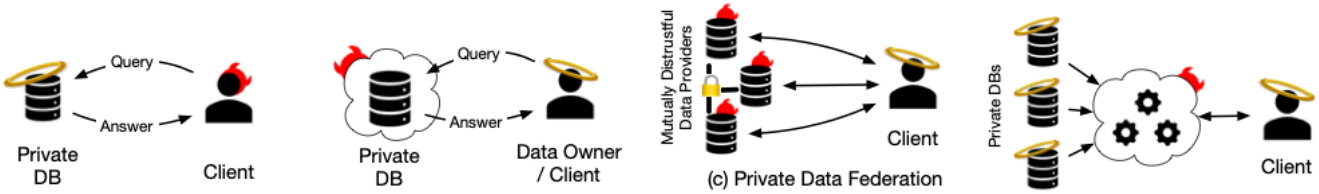


Figure 1: Trustworthy DBs in the query lifecycle



((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	$\geq$ 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 holomor-

phic 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.