Privacy by Design

A Strategic Framework for Choosing Between Encryption and Noise

Prepared By:

Max Forrest (30301436)
Chirag Kumar (30273579)
Sukhmanveer Singh (30291188)
Karanveer Singh (30282286)

Course: ISEC 611
Private Data Management

Table of Contents

Introduction

The Privacy Paradox in Modern Data Analytics

Companies today face a tough challenge. They need to use data to make smart business decisions but they also must protect people's personal information. Data has become incredibly valuable and this creates real tension. Laws like General Data Protection Regulation (GDPR), California Consumer Privacy Act (CCPA) and the Health Insurance Portability and Accountability Act(HIPAA) set strict rules on how personal data can be collected, processed, stored and shared [8]. Breaking these rules means big fines and damaged reputation. The problem isn't just following the law, it's figuring out how to analyze sensitive data safely [8].

The Computational Privacy Gap

Traditional approaches to data protection, such as access controls, role-based permissions, and encryption at rest, have long served as organizational security cornerstones [1]. However, these mechanisms share a critical vulnerability: they protect data only in its static state [1]. Once data must be decrypted and actively processed for analysis, it becomes exposed to a range of threats. This moment of decryption represents a significant window of vulnerability—potential breaches, insider threats, unauthorized access by administrative personnel, and compromise through side-channel attacks all become possible. This creates what we term the computational privacy gap: the inherent risk that emerges when sensitive data transitions from protected storage to active processing [1].

Privacy-Preserving Technologies: An Emerging Solution

To bridge this gap, organizations are increasingly turning to Privacy-Preserving Technologies (PPT)- specialized cryptographic and statistical techniques designed to enable computation and analysis on sensitive data without exposing the underlying information [1]. These approaches represent a fundamental shift in security philosophy, instead of trying to prevent unauthorized access to decrypted data, PPTs operate on the principle that the data need not be decrypted at all during processing [1].

However, the landscape of available PPTs is diverse and selecting the appropriate technology for a specific use case remains a complex decision. Different approaches offer different guarantees, performance characteristics and practical constraints [5]. Without clear guidance on when to use one approach versus another, practitioners often struggle to choose between these approaches, as each offers distinct advantages, limitations, and trade-offs between privacy protection and data usability.

Fundamentals of Privacy-Preserving Technology

What are Privacy-Preserving Technologies?

Privacy-preserving technologies (PPT) are statistical and cryptographic methods for meaningful computation and analysis of sensitive data while providing mathematical or

cryptographic guarantees that individual privacy is protected [1]. Rather than preventing unauthorized access to data (traditional security approach), PPTs operate under a different principle: they allow authorized computational processes to occur without exposing the sensitive information underlying those computations [1].

Key Properties of Effective PPTs
A robust Privacy-preserving technology should demonstrate following characteristics:

Privacy Guarantees: The technology must provide formal, mathematically proven privacy guarantees [1]. These should be expressed in precise terms that allow practitioners to understand exactly what privacy claims the technology makes and under what threat models these claims hold [1].

Data Utility Preservation: Privacy protection must not render the data useless for analysis. Effective PPTs maintain sufficient data utility to enable meaningful analytics, though always with trade-off between privacy strength and accuracy [7].

Computational Feasibility: The technology must be practically implementable within reasonable computational and resource constraints[5].

Scalability: The technology must scale appropriately with growing datasets and organizational needs [6]. Overhead(computational, memory or communication) should grow in a meaningful, predictable manner as data volume increases [6].

Transparency and Auditability: The mechanisms should be transparent and potentially auditable, allowing organisations to understand what privacy guarantees are in place and verify compliance [1].

The Landscape of Privacy-Preserving Approaches

Statistical Approaches: Differential Privacy represents a statistical solution to the computational privacy problem. Rather than preventing access to data, DP works by introducing carefully calibrated noise into query results or datasets [3]. The noise ensures that individual records cannot be reliably identified in the output while maintaining overall data utility for aggregate statistics and analysis [4].

Cryptographic Approaches: Homomorphic Encryption represents a cryptographic solution to computational privacy problem. HE allows computations to be performed directly on encrypted data without requiring decryption [1, 2]. This means that an analyst, cloud service provider or any other entity can process sensitive information while never having access to plaintext data itself [1].

Hybrid Approaches: Recent research explored hybrid approaches combining both HE and DP [3]. Cost-benefit analysis of hybrid models shows optimization potential, while differential privacy can be harnessed from the inherent noise in approximate homomorphic encryption methods [4]. These hybrid approaches represent an emerging

frontier where organizations might gain benefits from both cryptographic and statistical privacy guarantees [8].

Common Acronyms and Terminology

Core Technologies

Acronym	Full Term	Definition
HE	Homomorphic Encryption	Cryptographic technique allowing computations directly on encrypted data without decryption
FHE	Fully Homomorphic Encryption	HE supporting arbitrary computations (addition and multiplication) on ciphertext
SHE	Somewhat Homomorphic Encryption	HE supporting limited numbers of operations before decryption becomes necessary
DP	Differential Privacy	Statistical technique adding controlled noise to query results or data to prevent individual identification
LDP	Local Differential Privacy	DP applied at individual data points before aggregation, providing privacy without trusting central aggregators
PPT	Privacy-Preserving Technology	Broad category of cryptographic and statistical privacy protection methods

Related Concepts and Regulatory Framework

Acronym	Full Term	Definition
GDPR	General Data Protection Regulation	European Union regulation governing personal data collection, processing, and sharing

CCPA	California Consumer Privacy Act	California state law granting consumers rights over personal data collection and use
HIPAA	Health Insurance Portability and Accountability Act	U.S. federal law protecting healthcare data privacy
PII	Personally Identifiable Information	Information that can directly or indirectly identify an individual
PET	Privacy-Enhancing Technology	Broader category encompassing PPTs and other privacy protection mechanisms
PIA	Privacy Impact Assessment	Formal process evaluating how systems or policies affect individual privacy

Technical and Mathematical Terms

Term	**Definition**	**Context**
Ciphertext	Encrypted data resulting from encryption	Homomorphic Encryption: computations occur on ciphertext without decryption
Plaintext	Original unencrypted data	The input to encryption algorithms or the output of decryption
Noise	Statistical perturbation added to data or query results	Differential Privacy: noise magnitude determines privacy guarantee strength
Epsilon (ε)	Privacy loss parameter in differential privacy	Smaller ε = stronger privacy guarantee but lower data utility
Delta (δ)	Failure probability parameter in differential privacy	Acceptable probability that privacy guarantee may be breached

Query Result	Output of a computational operation (sum, mean, count, etc.)	The data released to users after privacy-preserving processing
Granularity	Level at which privacy protection applies	Record-level, query-level, or dataset-level protection
Utility	Usefulness and accuracy of data for analysis	Privacy-utility trade-off: stronger privacy often reduces analytical utility

Homomorphic Encryption (HE)

What is Homomorphic Encryption?

Homomorphic encryption (HE) is a cryptographic technique which allows computations to be performed directly on the encrypted data without requiring decryption and it also ensures that sensitive information remains protected throughout the analysis process [9, 10]. Unlike traditional encryption methods that only protect data at rest or in transit, HE maintains confidentiality even during active processing and computation [14]. This fundamental capability addresses a critical vulnerability data security: the risk that arises when encrypted data must be decrypted for analysis, a limitation extensively documented in privacy-preserving literature [12, 13].

The concept of homomorphic operations emerged as early as 1978 [9], but the first viable fully homomorphic encryption scheme was not realized until Craig Gentry's groundbreaking work at Stanford in 2009 [10]. Gentry's contribution, which utilized ideal lattices to prove that FHE was mathematically achievable, marked a pivotal moment in cryptography and transformed FHE from a theoretical concept into a practical reality [11].

Types of Homomorphic Encryption and Supported Operations

HE schemes are categorized into three types based on the operations they support and the computational depth they allow [11]:

Partially Homomorphic Encryption (PHE) supports only a single type of mathematical operation. Can perform either addition or multiplication but can perform that operation an unlimited number of times on encrypted data [15]. Classical examples include RSA encryption, which is partially homomorphic for multiplication [16], and Paillier encryption, which is partially homomorphic for addition [17]. PHE is the most computationally efficient variant but has limited practical applications due to its restricted operation set.

Somewhat Homomorphic Encryption (SHE) supports both addition and multiplication operations on encrypted data, but only for a limited number of computations before noise accumulation degrades the ciphertext quality [10]. As documented in practical applications research, SHE schemes must set a circuit depth limit prior to encryption, which presents challenges since many real-world computations have arbitrary or unknown depths. This limitation makes SHE suitable for applications with predictable, bounded computational requirements [12].

Fully Homomorphic Encryption (FHE) supports unlimited operations of both addition and multiplication on encrypted data, enabling arbitrary computations of any complexity. FHE represents the "holy grail" of homomorphic encryption, as it can theoretically perform any computation that could be done on plaintext data [10]. The most prominent FHE schemes include Brakerski-Gentry-Vaikuntanathan (BGV) [18], Brakerski/Fan-Vercauteren (BFV) [19], and Cheon-Kim-Kim-Song (CKKS)[20], each optimized for different computational contexts and data types.

Major FHE Schemes: BGV, BFV, CKKS

BGV (Brakerski-Gentry-Vaikuntanathan) is optimized for integer arithmetic with known circuit depth and provides fine-grained noise control through modulus switching [18]. BGV is best suited for applications requiring exact computation where the computational depth can be determined in advance.

BFV (Brakerski/Fan-Vercauteren) encodes messages in the most significant bits and uses a fixed modulus approach that simplifies parameter selection and implementation [19]. BFV performs well for batched operations on binary or small integer data and offers simpler implementation compared to BGV.

CKKS (Cheon-Kim-Kim-Song) is specifically designed for approximate arithmetic on real and complex numbers, making it particularly well-suited for machine learning, statistical computations, and scientific computing applications [20]. CKKS benefits from simpler approximate bootstrapping compared to the exact bootstrapping required by BGV and BFV, enabling more efficient handling of deep circuits.

Supported Operations

The fundamental operations supported by HE depend on the scheme type, but generally include:

Arithmetic operations: Addition, subtraction, multiplication of encrypted values [18]

Statistical computations: Sum, mean, variance, and other aggregate functions on encrypted datasets

Comparative operations: Some schemes support comparison operations (greater than, less than, equality checks), though these are computationally expensive

Machine learning inference: Neural network predictions, linear regression, and classification on encrypted data [20]

Performance and Computational Cost

Performance represents one of the most significant challenges for HE adoption, as the computational overhead of operating on encrypted data is substantially higher than traditional plaintext operations.

Computational Complexity: While basic operations on plaintext data may take microseconds, equivalent operations on encrypted data can take seconds or even minutes, depending on the encryption scheme and computational complexity. According to benchmarking research, FHE is particularly computationally expensive, with the computational cost per operation being thousands of times greater than plaintext computation.

Key Performance Factors

Ciphertext size: Encrypted data is significantly larger than plaintext, often consuming kilobytes per value compared to bytes for plaintext. This substantial size increase impacts both memory requirements and network bandwidth for data transmission.

Noise accumulation: Each homomorphic operation adds computational noise to the ciphertext [22]. When noise exceeds a certain threshold, the encrypted data cannot be correctly decrypted. FHE addresses this through "bootstrapping," a process that refreshes ciphertexts by homomorphically evaluating the decryption circuit to reduce noise, but bootstrapping itself is extremely computationally intensive and represents a significant performance bottleneck.

Hardware requirements: HE operations require substantial processing power and memory, making implementation costly and potentially limiting adoption to organizations with significant computational resources. Research on cost-benefit analysis indicates that specialized hardware acceleration can improve performance but adds to deployment costs [23].

Performance Improvements: Systematic benchmarking of HE libraries has shown that performance has been steadily improving through algorithmic optimizations, batching techniques (performing multiple operations simultaneously on packed ciphertexts), and hardware acceleration using specialized processors. However, even with these improvements, HE remains orders of magnitude slower than plaintext computation.

Data Utility and Efficiency

Unlike Differential Privacy which deliberately adds noise to protect privacy, HE aims to preserve exact data utility and produce computationally accurate results. When properly implemented with appropriate parameter selection, HE operations yield the same mathematical results as if the computation had been performed on plaintext data.

Accuracy Considerations

Deterministic results: HE produces deterministic outputs - encrypting the same plaintext with the same key produces the same ciphertext, and homomorphic operations yield exact results without statistical approximation in schemes like BGV and BFV [18][19]. This semantic security property ensures that identical plaintexts are indistinguishable after encryption.

Noise-induced errors: The primary threat to data utility in HE is noise accumulation during computation. If noise grows too large before bootstrapping or decryption, decryption errors occur, producing incorrect results. Research on noise-resilient HE frameworks has focused on managing this challenge through careful parameter selection and noise monitoring [21].

Precision limitations: Some HE schemes, particularly approximate HE variants like CKKS, work with fixed-point arithmetic and may introduce small rounding errors, but these are typically negligible for most applications and acceptable in contexts where approximate results are sufficient [20].

Trade-off with Performance: While HE preserves data utility better than statistical methods like DP, this comes at the cost of computational efficiency. Organizations must balance the need for exact results against the substantial performance overhead, a trade-off that depends heavily on the specific computational requirements and data sensitivity.

Security Analysis

Privacy Guarantees

HE provides the strongest form of privacy protection available among privacy-preserving technologies, rooted in cryptographic hardness rather than statistical obfuscation.

Cryptographic Foundations (Cryptographic Security)

HE's security is based on well-established mathematical problems such as the Learning With Errors (LWE) problem and the hardness of lattice-based problems like the Shortest Vector Problem (SVP) and Gap-SVP[26]. These problems are conjectured to be hard for both classical and quantum computers, making HE a promising candidate for post-quantum cryptography. Brakerski and Vaikuntanathan proved that leveled FHE can be based on the hardness of $O(n^{1.5+\varepsilon})$-approximation for lattice problems under quantum reductions, matching the security assumptions of standard lattice-based public-key encryption [25].

Key Management

Key Privacy Properties

Confidentiality during computation: Data remains encrypted throughout processing, meaning that even parties performing computations on the data cannot access the underlying plaintext. This enables "zero-trust" architectures where sensitive data, such as genomic information, can be processed in untrusted environments like public cloud platforms [14].

Protection against multiple attack vectors: HE defends against chosen plaintext attacks, side-channel attacks based on statistical analysis, and machine learning-based inference attacks. The randomized encryption ensures that identical plaintexts encrypt to different ciphertexts, preventing pattern recognition and frequency analysis attacks.

Semantic security: HE schemes provide semantic security (also known as IND-CPA security), meaning that an adversary cannot distinguish between the encryptions of any two messages of their choice, even with access to the public key and evaluation keys. This strong security guarantee is formally proven based on the underlying lattice problem hardness assumptions [18].

Key Management Requirements: The security of HE depends critically on secure key management. The secret decryption key must be protected from unauthorized access - if an adversary obtains the key, all privacy guarantees are lost. In multi-party scenarios, advanced techniques like multiparty homomorphic encryption (MHE) can distribute trust across multiple parties, eliminating single points of failure [24].

Scalability and Limitations

Despite its strong security guarantees, HE faces several limitations that affect real-world deployment:

Computational Overhead: The most significant limitation is performance. HE operations are orders of magnitude slower than plaintext computation, creating unacceptable latency for real-time applications. Research indicates that even with state-of-the-art implementations and hardware acceleration, the performance gap remains substantial [27]. This makes HE impractical for time-sensitive use cases such as real-time fraud detection or interactive analytics.

Limited Functionality: While FHE theoretically supports arbitrary computations, practical implementations often require specific types of operations or limit the complexity of functions that can be computed efficiently. Complex operations like division, modulo arithmetic, and certain comparison operations are difficult or impossible to implement efficiently. The computational cost increases exponentially with circuit depth, limiting the complexity of operations that can be performed practically [20].

Scalability Challenges: As dataset size grows, both computational time and memory requirements increase substantially. Research on protecting genomic data with HE demonstrates that computational cost is highly dependent on the specific data characteristics and processing requirements [6]. Processing large datasets with HE can require specialized hardware and significant infrastructure investment.

Lack of Standardization: There are numerous HE schemes and implementations with no widely accepted standard, leading to compatibility issues between different systems. This fragmented ecosystem makes it challenging for organizations to select appropriate solutions or integrate HE with existing infrastructure.

Expertise Requirements: Implementing HE correctly requires deep cryptographic expertise. Misconfigurations in key management, parameter selection (such as lattice dimension and modulus choice), or noise management can compromise either security or functionality.

Real-World Use Cases

Despite its limitations, HE is the preferred technology for specific high-value scenarios where absolute data confidentiality is paramount:

Healthcare and Genomics: Healthcare providers can analyze encrypted patient records across multiple institutions without compromising HIPAA compliance. Genomic research involving sensitive genetic data benefits from HE's ability to perform complex analytics while maintaining patient privacy. Research has demonstrated the feasibility of using HE for protecting genomic data in systems like i2b2, enabling collaborative research on encrypted medical information [6]. A recent study confirmed that FHE maintains privacy without compromising computational efficiency when handling encrypted genomic datasets for genotype report generation [28].

Financial Services: Banks use HE for secure credit scoring, fraud detection, and risk assessment while keeping customer financial data encrypted in cloud environments [14]. This enables financial institutions to leverage cloud computing's scalability while maintaining regulatory compliance and customer confidentiality.

Cloud Computing: HE enables secure outsourcing of data processing to untrusted cloud service providers. Organizations can leverage cloud computing resources for complex analytics without exposing proprietary or sensitive data to the cloud provider, a capability particularly valuable given the increasing adoption of cloud infrastructure.

Secure Multi-Party Computation: When multiple organizations need to jointly analyze data without revealing their individual datasets, HE provides a cryptographic foundation for secure collaboration. This is particularly valuable in supply chain analytics, consortium research, and regulatory reporting scenarios where data sharing is necessary but privacy concerns limit direct exchange of plaintext data [24].

Privacy-Preserving Machine Learning: HE enables AI/ML model training and inference on encrypted data, allowing organizations to build predictive models without accessing raw training data. This is critical for applications involving sensitive personal information where data sharing is legally restricted or organizationally unacceptable. Research on integrating HE with machine learning continues to advance, with CKKS showing particular promise for neural network computations [20].

Differential Privacy (DP)

What is Differential Privacy?

Differential Privacy (DP) is not about hiding data, rather it's about limiting what can be learned about any one individual from the results of an analysis. Instead of protecting raw data through encryption, DP accepts that analysts will see outputs, but ensures those outputs are safe to share. The key idea is that if the result of an analysis looks almost the same whether or not a specific person's data is included, then that person's privacy is protected. This idea is formalized through a mathematical guarantee that bounds how much any single record can influence the final result.

Formal Definition:

Differential Privacy (DP) is a statistical privacy framework that provides formal, mathematical guarantees that the inclusion or exclusion of any single individual's data has a bounded and provably small effect on the output of an analysis. Unlike cryptographic approaches such as Homomorphic Encryption, DP does not attempt to keep data confidential during computation. Instead, it ensures that released outputs do not leak sensitive information about individuals, even to adversaries with strong background knowledge [29, 30].

Mathematical Definition:

A randomized mechanism 'M' with domain $\mathbb{N}|x|$ is (ε,δ)-differentially private if for all $S \subseteq$ Range(M) and for all $x,y \in \mathbb{N}|x|$ such that $||x-y||1 \leq 1$:

$$\Pr[M(x) \in S] \leq e^{\varepsilon} \Pr[M(y) \in S] + \delta$$

Where the probability space is over the coin flips of the mechanism 'M'. If $\delta=0$, we say that 'M' is ε-differentially private [29]. The worst-case interpretation is that an attacker can only increase their odds by at most $(e\varepsilon)$of distinguishing the presence or absence of any individual under the most adverse conditions.

Terms:

- M - A randomized mechanism (algorithm) that takes a dataset as input and produces a privatized output.
- x,y - Neighboring datasets that differ in the data of at most one individual.
- $S \subseteq$ Range(M) - Any measurable subset of possible outputs of the mechanism M.

- Pr - Probability taken over the internal randomness (coin flips) of the mechanism M.
- E - The privacy budget controlling the strength of the privacy guarantee.
- δ - A small failure probability allowing the privacy guarantee to be violated with probability at most δ.

Privacy Guarantees and Threat Model

DP provides privacy guarantees that are independent of the attacker's computational power or prior knowledge, a key distinction from many traditional anonymization techniques, which often fail under. Linkage attacks or auxiliary information. The guarantee holds even if the adversary knows all records in the dataset except one [29].

Key properties of DP include:

- **Record-level protection**: Privacy guarantees apply at the level of individual data records.
- **Post-processing immunity**: Any function applied to DP outputs remains differentially private. Once data is differentially private, anything you do to the output cannot make it less private. This makes DP especially attractive for real-world analytics pipelines where outputs are reused, visualized, or shared [32].
- **Composable privacy loss**: Privacy degrades predictably as multiple queries are answered. When multiple DP mechanisms are applied sequentially, the overall privacy guarantee degrades in a quantifiable, predictable manner [36]. Sequential composition allows practitioners to track cumulative privacy loss across multiple queries or analyses.

These properties make DP particularly well-suited for repeated statistical queries, dashboards, and large-scale analytics systems, where outputs are released over time [32].

Noise Injection Mechanisms

Differential Privacy achieves its guarantees by introducing carefully calibrated random noise into query results or datasets. The amount and distribution of noise depend on the query's sensitivity and the desired privacy level.

The most commonly used mechanisms include:

1. **Laplace Mechanism:**
   Adds Laplace distributed noise proportional to the L1- sentivity of the query. It is widely used for numeric queries such as count, sum, and means [30].
   The magnitude of the noise is proportional to the query's sensitivity (the maximum amount any single record can change the output) divided by ε. For every function 'f' with sensitivity Δf:

$$M(D) = f(D) + Lap(\Delta f/ \varepsilon)$$

The Laplace mechanism is particularly suited for queries with unbounded output ranges [7]. Its symmetric, exponentially-decaying probability distribution ensures that larger noise values become increasingly unlikely while maintaining the required privacy guarantee.

2. **Gaussian Mechanism:**
   Adds Gaussian noise calibrated to L2 -sensitivity and supports (ε,δ)-DP. It is preferred in many modern systems due to better composability properties and in machine learning applications with gradient-based optimization algorithms [31]. The noise scale is determined by:

$$M(D) = f(D) + N(0, \sigma^2) \text{ where } \sigma^2 \geq 2\ln(1.25/\delta)(\Delta f/\varepsilon)^2$$

   This mechanism typically requires larger noise magnitudes than the Laplace mechanism for equivalent privacy guarantees but offers analytical advantages in certain contexts.
   In all cases, f(D) denotes the true (non-private) query result, while M(D) denotes the randomized, differentially private output released to users.
3. **Exponential Mechanism:**
   It is used for non-numeric outputs such as selecting items or categories while preserving privacy [29].

The key point is that DP noise is deliberate and controlled, not an implementation artifact or approximation error [30, 31]. In practice, most DP analytics pipelines rely on Laplace or Gaussian mechanisms for aggregate statistics.

Supported Operations

DP works best with aggregate and statistical computations:

- Counts and frequencies
- Sums and means
- Histograms
- Trends over large populations
- Percentiles and quantiles (with approximation)
- Linear regression and other machine learning models

DP has been successfully integrated into training algorithms for neural networks, decision trees, and clustering through techniques like DP-SGD (Differentially Private Stochastic Gradient Descent) [42]. Abadi et al. demonstrated that deep learning models can achieve high accuracy while providing strong privacy guarantees through careful noise calibration during training. Techniques exist for DP-compliant regression analysis,

hypothesis testing, and even synthetic data generation that preserves statistical properties of the original dataset [43].

More complex operations are possible but typically require careful sensitivity analysis and may suffer from reduced accuracy as noise accumulates [33]. Importantly, DP does not preserve exact values. All results are probabilistic approximations whose accuracy depends on dataset size, query sensitivity, and privacy parameters.

DP Implementation Models

Local Differential Privacy:

In local differential privacy, noise is added to individual records before data collection, ensuring that even the data collector cannot access raw sensitive information [34]. This trust model is particularly valuable when users cannot trust the data aggregator. Apple's implementation in iOS devices exemplifies LDP, where noise is added on-device before transmission [37].

Advantages: No trusted curator required, strong protection against data breaches.

Limitations: Requires significantly more noise than central DP for equivalent utility; suitable primarily for simple statistical queries [38].

Central Differential Privacy:

Central DP assumes a trusted curator who receives raw data and adds noise to query outputs or intermediate computations [39]. This model enables more complex analyses with better accuracy-privacy tradeoffs than Local DP.

Advantages: Superior data utility for equivalent privacy budgets, supports complex analytical queries.

Limitations: Requires trust in the central curator, vulnerable to insider threats [40].

Differential Privacy Parameters and the Privacy-Utility Trade-off

The privacy guarantee in DP is controlled primarily by the parameter Epsilon (ε) and is referred to as the privacy budget.

Smaller ε ⇒ stronger privacy, more noise, lower accuracy of data

Larger ε ⇒ weaker privacy, less noise, higher accuracy of data

The δ parameter represents the probability of catastrophic failure and is typically set to be extremely small (less than 1/n where n is dataset size) [31].

There is no universally ‘correct’ ε. Contextual factors (user intent, data use-case, adversary knowledge, regulatory setting) cannot be encapsulated in ε alone. Choosing ε is a policy and risk-management decision, not purely a technical one. Hsu et al. argue

that ε should be chosen based on risk tolerance, regulatory context, and business value, not abstract theory [7].

Privacy Budget

Privacy Budget is a quantitative measure of the maximum allowable privacy loss when analyzing a dataset. It defines an upper bound on how much an individual's data can affect the outcome of queries.

A critical operational consideration in DP systems is privacy budget. Each query consumes part of the total privacy budget, and repeated queries degrade privacy guarantees over time.

This makes DP particularly suitable for controlled, well-defined analytics pipelines, but more challenging for open-ended exploratory analysis [32]. Systems must track cumulative privacy loss over time. Budget accounting is one of DP's biggest operational challenges, but also one of its strengths, because it makes privacy loss explicit and auditable [32].

Organizations face critical decisions including:

- How to distribute ε across multiple queries?
- Whether to reserve budget for future analysis or exhaust it immediately?
- How to balance stakeholder needs and different analytical requirements within a fixed privacy budget?

Efficiency and Scalability

DP exhibits a favorable relationship between data size and utility. As dataset size increases, the relative impact of noise decreases, leading to improved accuracy for the same privacy budget. This makes DP particularly effective for large population datasets, repeated aggregate reporting and public data releases.

Advantages of DP over HE

- Near-plaintext computational cost
- Minimal memory overhead
- No ciphertext expansion
- No cryptographic keys are required

Another advantage of DP over cryptographic approaches is computational efficiency. Noise addition is typically a lightweight operation (microseconds per query), and DP systems scale well to very large datasets. As a result, DP is widely deployed in production systems at scale, including by Google, Apple, and the U.S. Census Bureau [34, 35]. In contrast to HE, DP benefits directly from scale rather than being constrained by it [33].

For a dataset of size n and a count query, the relative error decreases proportionally to $1/\sqrt{n}$. Conversely, small datasets or queries returning small counts suffer from poor utility under strong privacy guarantees. This presents challenges for analyzing rare events or minority populations.

Data Utility and Accuracy

DP introduces intentional error into outputs. The accuracy depends on several factors like the dataset size, query sensitivity, privacy budget and number of queries.

Individual-level insights are unreliable. For small datasets or high-sensitivity queries, noise can significantly distort results. For large datasets, the noise often becomes negligible relative to the signal [30]. Unlike HE, DP does not aim for exact correctness. Instead, it provides bounded statistical error with formal privacy guarantees. For many real-world analytics tasks, this is acceptable and often preferable but for tasks that require deterministic outputs, DP is fundamentally the wrong tool [29].

The utility of DP mechanisms is typically measured through:

- **Absolute error**: The expected magnitude of noise added to query outputs.
- **Relative error**: Noise magnitude as a proportion of the true answer, particularly relevant for small counts.
- **Statistical validity**: Whether DP-protected results enable valid statistical inference and hypothesis testing.

Strengths and Limitations of DP

Strengths of DP include strong formal privacy guarantees, excellent scalability and performance, low infrastructure and operational cost, well supported tooling and growing standards, increasing regulatory acceptance, etc. DP is best suited for trend analysis, aggregate reporting, population-level insights, etc

Limitations of DP include accuracy loss due to noise, complex privacy budget management, etc. It is poorly suited for individual-level decisions, exact auditing or reconciliation, applications requiring deterministic outputs, etc. These limitations are fundamental to the DP model and not merely implementation artifacts [29]. Determining query sensitivity can be computationally intensive for complex queries, sometimes requiring worst-case analysis across all possible datasets [41].

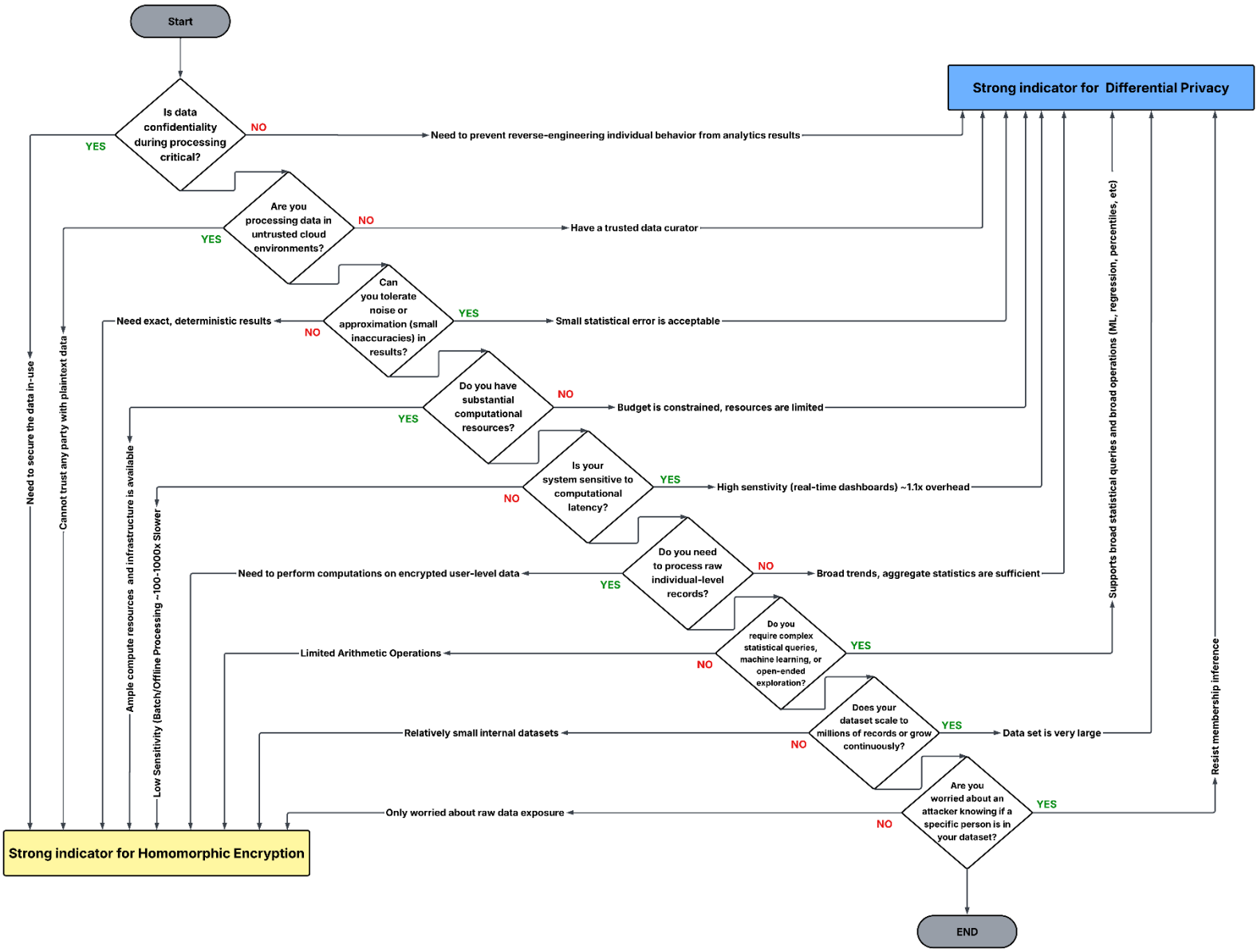
Misconfigured DP systems particularly those with poorly chosen ε values without understanding their implications can lead to systems that appear private but offer weak protection, or systems that are so noisy they are useless [7]. Complex non-linear operations may require custom mechanism design or result in substantial accuracy loss.

When to Use Differential Privacy over Homomorphic Encryption

DP is the preferred approach when:

- Aggregate statistics are sufficient
- Large datasets are available
- Results can tolerate approximation
- Real-time or interactive analytics are required
- Computational resources are constrained
- Data is processed in trusted environments

Decision Framework



Framework Use Cases

Census Data

The US Census Bureau collects data regarding households in the country every decade, publishing the results for redistricting and research purposes. Providing adequate privacy to US Census data is a difficult task given the range of use cases those data are used for. To avoid violating the privacy of respondents while still maintaining the ability to request useful data by external data users, privacy-preserving technologies are required. The Census Bureau uses a Disclosure Avoidance System (DAS) to protect private data while publishing useful results for further study and data processing.

Data Confidentiality

Confidentiality of data during processing is important but is less of a priority than the identifiability of individual records.

Data Environment

For the DAS, data is processed using Amazon Web Services (AWS). While processing data on the public cloud is a strong indicator that Homomorphic Encryption would be the appropriate PPT, the US Census Bureau instead makes use of AWS GovCloud [44] which provides increased information security by restricting access to government organizations and approved private entities and adhering to more strict security standards than a traditional deployment [45].

Noise Tolerance

Over 90 million households were included in the 2020 Census. The scale of this dataset means that for most use cases, adding sufficient noise to the query results to provide adequate privacy has only a minor impact on the utility of the data. Minority groups will be disproportionally impacted by the added noise, however masking their data is an intended feature by the Census Bureau [46].

Computational Resources

According to Garfinkel, when testing the “TopDown” algorithm for implementing Differential Privacy in the Census Bureau’s DAS, using AWC clusters for computation [47], “The cost for each run was typically between $100 and $10,000, depending on which part of algorithm was being tested and how much test data was being used.” He went on to explain that the compute time for the testing phase of implementation cost several million dollars. The expected costs for a homomorphic encryption solution would be significantly higher [33], making it a poor choice for this use-case.

Latency Sensitivity

The processing of Census data is not overly sensitive to latency, however the volume of data processed is sensitive to efficiency.

Individual-level Records

Data published by the US Census Bureau is primarily utilized to aggregate statistics and analyze larger trends. User-level records must be obfuscated for data-users and therefore noise is more relevant than encryption.

Required Operations

Census data is utilized in complex queries, not just simple mathematical operations, making HE a non-ideal choice of a PPT.

Dataset Size

This dataset includes records for over 90 million households and was made public, making it especially suitable for protection using DP.

Identifiability

Being able to identify whether a particular data owner's records exist within the Census data would fail to meet the goals of the US Census Bureau [46].

Conclusion

As the indicators in our decision framework overwhelmingly support the use of Differential Privacy instead of Homomorphic Encryption in the case of the US Census data, we would agree with the Census Bureau's decision to utilize DP as their PPT of choice.

Hospitals and Healthcare Data

Large hospital systems are required to publicly report quality metrics including hospital-acquired infection rates, readmission rates, mortality rates, and patient satisfaction scores to regulatory bodies such as the Centers for Medicare & Medicaid Services (CMS). These reports inform public health policy, hospital accreditation, and patient decision-making, but must protect individual patient privacy while maintaining statistical validity.

Healthcare data is uniquely sensitive due to the presence of personally identifiable health information, strict regulatory oversight (HIPAA, etc.), and the need for both high accuracy and timely insights. Use cases range from clinical analytics and population health studies to billing, fraud detection, and research collaborations.

Data Confidentiality

Confidentiality during processing is secondary to preventing re-identification in published outputs. Hospital IT systems are HIPAA-compliant with existing security controls, making DP's output protection more relevant than HE's encryption during computation.

Data Environment

Hospital data is processed within HIPAA-compliant environments (Azure Healthcare, AWS HIPAA services) with trusted IT staff and Business Associate Agreements. The trusted curator model applies, eliminating HE's primary benefit of enabling computation by untrusted parties.

Noise Tolerance

Hospital quality metrics can tolerate 5-10% error. With 50,000 patient encounters and $\varepsilon = 1.0$, a true infection rate of 2.5% might be reported as 2.3%-2.7%, sufficient for CMS reporting and public transparency.

Computational Resources

DP implementations cost $200-500/month on standard infrastructure with near-instant query times. HE would require $5,000+/month, specialized hardware (256GB+ RAM), and 500-1,000× slower processing which is economically impractical for routine quality reporting.

Latency Sensitivity

Real-time clinical dashboards require sub-second response times for infection monitoring and safety alerts. DP adds negligible latency (<1%), while HE's minutes-to-hours delays would render dashboards unusable.

Individual-level Records

Quality metrics are exclusively aggregate statistics (mean length of stay, readmission rates). No individual patient records are released, making DP's query-level protection ideal and HE's record-level encryption unnecessary.

Required Operations

Hospital analytics require complex operations like percentile calculations, stratified analysis, regression models, and ML for risk stratification. DP supports these operations efficiently, while HE is limited to basic arithmetic.

Dataset Size

Large hospital systems process 50,000-500,000+ patient records annually. DP's accuracy improves with scale, while HE's computational overhead becomes prohibitive at these volumes.

Identifiability

Preventing membership inference attacks is critical for HIPAA compliance and patient trust. DP's mathematical guarantee (ε = 1.0 limits adversary advantage to factor of 2.7) directly addresses this requirement, protecting against re-identification even with background knowledge.

Conclusion

The overwhelming alignment with DP's strengths makes it the clearly appropriate privacy-preserving technology for hospital quality metrics reporting.

Consumer Analytics Platforms

Consumer analytics platforms (such as Google Analytics and Chrome telemetry) collect large-scale behavioural data to understand usage patterns, improve performance, detect abuse, and run experiments. These systems prioritize scale, low latency, and aggregate insights, while still needing to prevent re-identification of individual users.

Data Confidentiality

Users fundamentally distrust tech companies with raw behavioural data (browsing history, app usage). Local DP adds noise on-device before transmission, meaning the company never sees plaintext data and sees only noisy aggregates.

Data Environment

Data is generated on user devices (phones, browsers, laptops) in an untrusted environment. Local DP processes data client-side before sending to company servers, eliminating the need to trust Google, Apple, or any central party with raw behavioural information.

Noise Tolerance

Product analytics can tolerate 10-20% error for decision-making. With billions of users, Local DP (ε = 2-4) provides trends like '~30% of users enabled dark mode' with sufficient accuracy while maintaining strong privacy.

Computational Resources

Local DP requires minimal resources: noise addition happens in milliseconds on user devices, and server-side aggregation costs $100-500/month. HE would be prohibitively

expensive and computationally infeasible for billions of users generating continuous data streams.

Latency Sensitivity

Product teams need real-time dashboards for A/B testing and feature rollouts. Local DP adds negligible overhead (microseconds for noise generation), enabling hourly metric updates, while HE's processing delays would make real-time analytics impossible at this scale.

Individual-level Records

Analytics focus exclusively on aggregate trends ('% of users who clicked X') rather than individual behavior tracking. Local DP perfectly aligns with this by ensuring companies only receive noisy aggregates, never individual records.

Required Operations

User analytics involve diverse operations: feature usage counts, session duration histograms, click-through rates, and ML-based recommendation models. Local DP supports these statistical queries efficiently, while HE cannot scale to billions of users or complex analytics workflows.

Dataset Size

Google Chrome has ~3 billion users, Apple iOS has ~1.5 billion devices. At this scale, Local DP's noise becomes negligible relative to signal, making it ideal for population-level insights while HE's overhead would be computationally impossible.

Identifiability

Preventing membership inference attacks is critical since users must not be identifiable in analytics datasets. Local DP's on-device noise ($\varepsilon = 2\text{-}4$) provides plausible deniability, and companies literally cannot reveal individual data in response to legal requests because they never possess it.

Conclusion

Differential Privacy is the overwhelmingly appropriate technology for large-scale user analytics, as demonstrated by its deployment at Google (Chrome, Android) and Apple (iOS, macOS) serving billions of users.

Financial Data (Insurance Companies)

Insurance companies collect detailed customer data - policy numbers, claims history, credit scores, health/medical information and financial details for actuarial modeling, premium pricing and regulatory compliance. This data is highly sensitive due to PII, financial information and strict regulations (SOX, PCI-DSS, GDPR). Privacy-preserving

technologies are essential when outsourcing risk calculations to cloud providers or third-party actuaries while maintaining perfect confidentiality.

Data Confidentiality

Customer financial and health data requires absolute confidentiality during processing. Insurance companies often use cloud platforms for scalability, making HE ideal for "compute-without-reveal" scenarios where actuaries process encrypted customer records without decryption.

Data Environment

Insurance analytics frequently run on public cloud platforms (AWS, Azure) or third-party actuarial services. Unlike Census data's GovCloud, insurance cloud deployments are typically commercial environments where HE provides critical protection against honest but curious providers accessing raw customer data.

Noise Tolerance

Individual-level risk scoring and premium calculations require exact mathematical results. Adding DP noise would create pricing inaccuracies or unfair premiums. HE preserves 100% accuracy which is critical when a $0.01 pricing error across millions of policies costs millions.

Computational Resources

HE compute costs are substantial. Actuarial modeling for 1M policies might require GPU clusters costing $1000-$10000 per run(similar to Census DAS costs but for smaller datasets. Large insurers with existing ML infrastructure can absorb this; smaller firms cannot.

Latency Sensitivity

Risk model updates and pricing table generation are batch processes(overnight/weekly). Real-time fraud detection needs milliseconds (DP territory), but actuarial modeling tolerates hours of HE processing.

Individual-level Records

Core insurance analytics operate on individual policyholder records- calculating personalised risk scores, premiums and reserves. HE enables exact computations on encrypted per-customer data unlike Census aggregates where DP suffices.

Required Operations

Insurance requires exact arithmetic (multiplication for risk weighting, addition for portfolio totals). HE's BGV/BFV schemes excel here. Complex statistical exploration better suits DP, but insurance prioritizes deterministic financial calculations.

Dataset Size

Typical insurance portfolios range from 100K-10M policies. HE scales reasonably for this range on GPU infrastructure. Census-scale (90M+ households) overwhelms HE memory/compute requirements.

Identifiability

Primary Threat: raw data exposure during cloud processing, not membership inference from aggregates. Attackers seek individual policyholder records (SSN, claims history), which HE prevents through perfect encryption. DP protects published statistics from re-identification.

Conclusion

The decision framework strongly supports Homomorphic Encryption for insurance risk modeling. Perfect confidentiality during cloud/third-party processing, exact arithmetic requirements, individual-level computations and tolerance for batch latency align perfectly with HE strengths. While computational costs are high, large insurers treating analytics as a core competency can justify the investment. DP would be suboptimal due to unacceptable accuracy loss for pricing/actuarial work.

Cloud Computing

Healthcare organizations, financial institutions, and enterprises increasingly rely on cloud computing platforms for computationally intensive analytics and machine learning workloads. However, outsourcing sensitive data processing to third-party cloud providers creates significant privacy concerns.

Data Confidentiality

Cloud computing involves processing data on infrastructure owned by third parties such as AWS, Azure, or Google Cloud. For highly sensitive data including patient medical records, financial transactions, or proprietary research, any exposure of plaintext data to the cloud provider represents an unacceptable risk. This creates a strong indicator for Homomorphic Encryption.

Data Environment

By definition, public cloud platforms are untrusted environments. Organizations have no control over cloud provider employees, security practices beyond contracts, or potential government surveillance. HE ensures data remains encrypted throughout computation, enabling zero-trust architectures. This is another strong indicator for Homomorphic Encryption.

Noise Tolerance

Many cloud workloads require exact, deterministic results. Financial calculations for account balances and regulatory reporting cannot tolerate statistical noise. Healthcare diagnostics and clinical decision support require precise computations where errors could lead to incorrect treatment decisions. This strongly favors HE over DP.

Computational Resources

Organizations choosing cloud computing demonstrate willingness to invest in computational resources. Cloud platforms provide unlimited scalability through on-demand high-performance computing clusters. While HE operations are expensive, elastic cloud infrastructure allows organizations to provision resources as needed through batch processing and off-peak scheduling. For high-value data assets, the computational cost is justified by security benefits.

Latency Sensitivity

Many cloud workloads operate in batch mode: overnight financial reporting, weekly genomic analysis, monthly risk assessments. These can accommodate HE's processing times measured in hours rather than seconds. However, real-time applications like fraud detection or interactive dashboards requiring sub-second responses would not be suitable for HE.

Individual-level Records

Cloud applications frequently require individual record processing: customer

recommendations, personalized healthcare plans, individual credit scoring. HE excels at protecting individual-level data as each record remains encrypted, while DP is designed primarily for aggregate queries. Again, this is a strong indicator of HE.

Required Operations

Typical cloud analytics include operations well-suited to HE: statistical computations (means, sums), linear algebra for machine learning, and polynomial evaluations. Modern FHE schemes like CKKS efficiently support these.

Dataset Size

HE performs well on datasets from thousands to millions of records, covering medical research, regional financial services, and enterprise analytics.

Identifiability

Cloud computing scenarios often involve protecting data not just against direct access but also against sophisticated inference attacks. HE provides strong protection against such inference attacks because the cloud provider works exclusively with encrypted

ciphertexts that reveal no information about the underlying data. This cryptographic guarantee is significantly stronger than DP's statistical privacy, making HE the preferred choice for preventing identifiability in untrusted cloud environments.

Conclusion

Following the decision framework, cloud computing presents strong indicators for Homomorphic Encryption: critical confidentiality in untrusted environments, exact computation requirements, substantial resources, individual-level processing, and moderate dataset scales. Cloud computing represents one of the strongest applications for HE, directly addressing the fundamental challenge of outsourcing sensitive computations to third-party infrastructure.

References

[1] David W. Archer, Borja de Balle Pigem, Dan Bogdanov, Mark Craddock, Adria Gascon, Ronald Jansen, Matjaž Jug, Kim Laine, Robert McLellan, Olga Ohrimenko, Mariana Raykova, Andrew Trask, and Simon Wardley. 2023. UN Handbook on Privacy-Preserving Computation Techniques. https://doi.org/10.48550/ARXIV.2301.06167

[2] Practical Applications of Homomorphic Encryption: In Proceedings of the International Conference on Security and Cryptography, 2012. SciTePress - Science and Technology Publications, Rome, Italy, 5–14. https://doi.org/10.5220/0003969400050014

[3] Christoph Dobraunig, Lorenzo Grassi, Lukas Helminger, Christian Rechberger, Markus Schofnegger, and Roman Walch. 2023. Pasta: A Case for Hybrid Homomorphic Encryption. TCHES 2023, 3 (June 2023), 30–73. https://doi.org/10.46586/tches.v2023.i3.30-73

[4] Tabitha Ogilvie. 2024. Differential Privacy for Free? Harnessing the Noise in Approximate Homomorphic Encryption. In Topics in Cryptology – CT-RSA 2024, Elisabeth Oswald (ed.). Springer Nature Switzerland, Cham, 292–315. https://doi.org/10.1007/978-3-031-58868-6_12

[5] Jean Louis Raisaro, Gwangbae Choi, Sylvain Pradervand, Raphael Colsenet, Nathalie Jacquemont, Nicolas Rosat, Vincent Mooser, and Jean-Pierre Hubaux. 2018. Protecting Privacy and Security of Genomic Data in i2b2 with Homomorphic Encryption and Differential Privacy. IEEE/ACM Trans. Comput. Biol. and Bioinf. 15, 5 (September 2018), 1413–1426. https://doi.org/10.1109/TCBB.2018.2854782

[6] Charles Gouert, Dimitris Mouris, and Nektarios Georgios Tsoutsos. 2022. SoK: New Insights into Fully Homomorphic Encryption Libraries via Standardized Benchmarks. (2022), 139–161. Retrieved January 14, 2026 from https://eprint.iacr.org/2022/425

[7] Justin Hsu, Marco Gaboardi, Andreas Haeuerlen, Sanjeev Khanna, Arjun Narayan, Benjamin C. Pierce, and Aaron Roth. 2014. Differential Privacy: An Economic Method for Choosing Epsilon. https://doi.org/10.48550/arXiv.1402.3329

[8] Rezak Aziz, Soumya Banerjee, Samia Bouzefrane, and Thinh Le Vinh. 2023. Exploring Homomorphic Encryption and Differential Privacy Techniques towards Secure Federated Learning Paradigm. Future Internet 15, 9 (September 2023), 310. https://doi.org/10.3390/fi15090310

[9] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. 1978. On Data Banks and Privacy Homomorphisms. Foundations of Secure Computation 4, 11 (1978), 169–180.

[10] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In Proceedings of the forty-first annual ACM symposium on Theory of computing, May 31, 2009. ACM, Bethesda MD USA, 169–178. https://doi.org/10.1145/1536414.1536440

[11] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. 2010. Fully Homomorphic Encryption over the Integers. In Advances in Cryptology – EUROCRYPT 2010, Henri Gilbert (ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 24–43. https://doi.org/10.1007/978-3-642-13190-5_2

[12] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. 2011. Can homomorphic encryption be practical? In Proceedings of the 3rd ACM workshop on Cloud computing security workshop, October 21, 2011. ACM, Chicago Illinois USA, 113–124. https://doi.org/10.1145/2046660.2046682

[13] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A. Reuter, and Martin Strand. 2015. A Guide to Fully Homomorphic Encryption. Retrieved January 15, 2026 from https://eprint.iacr.org/2015/1192

[14] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. 2019. A Survey on Homomorphic Encryption Schemes: Theory and Implementation. ACM Comput. Surv. 51, 4 (July 2019), 1–35. https://doi.org/10.1145/3214303

[15] Caroline Fontaine and Fabien Galand. 2007. A Survey of Homomorphic Encryption for Nonspecialists. EURASIP J. on Info. Security 2007, 1 (December 2007), 013801. https://doi.org/10.1155/2007/13801

[16] R. L. Rivest, A. Shamir, and L. Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21, 2 (February 1978), 120–126. https://doi.org/10.1145/359340.359342

[17] Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Advances in Cryptology — EUROCRYPT '99, 1999. Springer, Berlin, Heidelberg, 223–238. https://doi.org/10.1007/3-540-48910-X_16

[18] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2012. (Leveled) fully homomorphic encryption without bootstrapping. In Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, January 08, 2012. ACM, Cambridge Massachusetts, 309–325. https://doi.org/10.1145/2090236.2090262

[19] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat Practical Fully Homomorphic Encryption. Retrieved January 18, 2026 from https://eprint.iacr.org/2012/144

[20] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic Encryption for Arithmetic of Approximate Numbers. In Advances in Cryptology – ASIACRYPT 2017, 2017. Springer International Publishing, Cham, 409–437. https://doi.org/10.1007/978-3-319-70694-8_15

[21] Navneet Agarwal, Sanat Anand, and Manoj Prabhakaran. 2019. Uncovering Algebraic Structures in the MPC Landscape. In Advances in Cryptology – EUROCRYPT 2019, 2019.

Springer International Publishing, Cham, 381–406. https://doi.org/10.1007/978-3-030-17656-3_14

[22] Léo Ducas and Daniele Micciancio. 2015. FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. In Advances in Cryptology -- EUROCRYPT 2015, 2015. Springer, Berlin, Heidelberg, 617–640. https://doi.org/10.1007/978-3-662-46800-5_24

[23] Cas Cremers, Samed Düzlü, Rune Fiedler, Marc Fischlin, and Christian Janson. 2021. BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. In 2021 IEEE Symposium on Security and Privacy (SP), May 2021. 1696–1714. https://doi.org/10.1109/SP40001.2021.00093

[24] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. 2012. Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE. In Advances in Cryptology – EUROCRYPT 2012, 2012. Springer, Berlin, Heidelberg, 483–501. https://doi.org/10.1007/978-3-642-29011-4_29

[25] Zvika Brakerski and Vinod Vaikuntanathan. 2011. Efficient Fully Homomorphic Encryption from (Standard) LWE. In 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science, October 2011. 97–106. https://doi.org/10.1109/FOCS.2011.12

[26] Oded Regev. 2009. On lattices, learning with errors, random linear codes, and cryptography. J. ACM 56, 6 (September 2009), 1–40. https://doi.org/10.1145/1568318.1568324

[27] Thomas Prantl, Simon Engel, Lukas Horn, Dennis Kaiser, Lukas Iffländer, André Bauer, Christian Krupitzer, and Samuel Kounev. 2023. Performance Impact Analysis of Homomorphic Encryption: A Case Study Using Linear Regression as an Example. In *Information Security Practice and Experience*, 2023. Springer Nature, Singapore, 284–298. https://doi.org/10.1007/978-981-99-7032-2_17

[28] Aadit Shah, Surindernath Sivakumar, and Prabakaran N. 2026. Encrypted intelligence: A comparative analysis of homomorphic encryption frameworks for privacy-preserving AI. Journal of Economy and Technology 4, (January 2026), 252–265. https://doi.org/10.1016/j.ject.2025.08.001

[29] Cynthia Dwork and Jing Lei. 2009. Differential privacy and robust statistics. In Proceedings of the forty-first annual ACM symposium on Theory of computing, May 31, 2009. ACM, Bethesda MD USA, 371–380. https://doi.org/10.1145/1536414.1536466

[30] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In Theory of Cryptography, Shai Halevi and Tal Rabin (eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 265–284. https://doi.org/10.1007/11681878_14

[31] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. Foundations and Trends® in Theoretical Computer Science 9, 3–4 (August 2014), 211–487. https://doi.org/10.1561/0400000042

[32] Frank D. McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, June 29, 2009. ACM, Providence Rhode Island USA, 19–30. https://doi.org/10.1145/1559845.1559850

[33] Kamalika Chaudhuri and Claire Monteleoni. 2008. Privacy-preserving logistic regression. In Advances in Neural Information Processing Systems, Neural Information Processing Systems Foundation (ed.). Curran, Red Hook, NY.

[34] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, November 03, 2014. ACM, Scottsdale Arizona USA, 1054–1067. https://doi.org/10.1145/2660267.2660348

[35] John M. Abowd. 2018. The U.S. Census Bureau Adopts Differential Privacy. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, July 19, 2018. ACM, London United Kingdom, 2867–2867. https://doi.org/10.1145/3219819.3226070

[36] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2017. The Composition Theorem for Differential Privacy. IEEE Trans. Inform. Theory 63, 6 (June 2017), 4037–4049. https://doi.org/10.1109/TIT.2017.2685505

[37] Differential Privacy Team. 2017. Learning with Privacy at Scale. Apple Machine Learning Research. Retrieved from https://machinelearning.apple.com/research/learning-with-privacy-at-scale

[38] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. 2018. Minimax Optimal Procedures for Locally Private Estimation. Journal of the American Statistical Association 113, 521 (January 2018), 182–201. https://doi.org/10.1080/01621459.2017.1389735

[39] Alexandra Wood, Micah Altman, Aaron Bembenek, Mark Bun, Marco Gaboardi, James Honaker, Kobbi Nissim, David R. O'Brien, Thomas Stienke, and Salil Vadhan. 2018. Differential Privacy: A Primer for a Non-Technical Audience. Vanderbilt Journal of Entertainment & Technology 21, (2018), 209–276.

[40] Ilya Mironov. 2012. On significance of the least significant bits for differential privacy. In Proceedings of the 2012 ACM conference on Computer and communications security, October 16, 2012. ACM, Raleigh North Carolina USA, 650–661. https://doi.org/10.1145/2382196.2382264

[41] Marco Gaboardi, Emilio Jesus Gallego Arias, Justin Hsu, Aaron Roth, and Zhiwei Steven Wu. 2014. Dual Query: Practical Private Query Release for High Dimensional Data. In Proceedings of the 31st International Conference on Machine Learning, June 18, 2014. PMLR, 1170–1178. https://proceedings.mlr.press/v32/gaboardi14.html

[42] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable Private Learning with PATE. https://doi.org/10.48550/ARXIV.1802.08908

[43] Moritz Hardt, Katrina Ligett, and Frank Mcsherry. 2012. A Simple and Practical Algorithm for Differentially Private Data Release. In Advances in Neural Information Processing Systems, 2012. Curran Associates, Inc., 2339–2347.

[44] 2020. U.S. Census brings nationwide count to the AWS Cloud | AWS Public Sector Blog. Retrieved January 29, 2026 from https://aws.amazon.com/blogs/publicsector/us-census-brings-nationwide-count-aws-cloud/

[45] Joseph Mahakian, Scott Holmdahl, Quadri Bada, Steffani Silva, and Zach Tretler. 2020. AWS GovCloud Resource and Cost Analysis. The MITRE Corporation. Retrieved January 29, 2026 from https://apps.dtic.mil/sti/html/trecms/AD1114483/

[46] U.S. Census Bureau, Disclosure Avoidance for the 2020 Census: An Introduction, U.S. Government Publishing Office, Washington, DC, November 2021