Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.087848

REVIEW

Privacy-Preserving Cryptography for Machine Learning: A Comprehensive Review of Homomorphic Encryption, Differential Privacy, Secure Computation, and Post-Quantum Foundations

Quang-Vinh Dang[1], Dat Le[2,*], Minh Ngoc Dinh[3] and Ngoc-Son-An Nguyen[4]

[1]School of Computing and Innovative Technologies, British University Vietnam, Hung Yen, Hanoi, Vietnam
[2]Department of Science, Technology, and International Projects, Ho Chi Minh City University of Economics and Finance, Ho Chi Minh City, Vietnam
[3]School of Data Science and Information Technology, Millennia Education, Ho Chi Minh City, Vietnam
[4]Faculty of Information Technology, Industrial University of Ho Chi Minh City, Ho Chi Minh City, Vietnam

*Corresponding Author: Dat Le. Email: datla@uef.edu.vn

Received: 24 June 2026; Accepted: 25 August 2026; Published: 22 October 2026

ABSTRACT: Deploying machine learning on sensitive data has made privacy a first-order design constraint. Over the past five years privacy-preserving machine learning (PPML) has matured from isolated proofs of concept into an ecosystem of cryptographic and statistical techniques, each occupying a distinct point in the trade-off space among confidentiality, integrity, utility, cost and trust. This review surveys that ecosystem for a research audience. We examine homomorphic encryption, including several currently available bootstrapping styles and encrypted transformer inference; differential privacy in deep learning, analytics and the private fine-tuning of large language models; privacy budget management through composition theorems, numerical accountants and empirical auditing; secure multi-party computation; zero-knowledge proofs as an integrity complement to confidentiality; federated learning, covering secure aggregation, gradient-leakage attacks, verifiable aggregation and federated adaptation of large models; encrypted search and its leakage-abuse attacks; and post-quantum foundations following the 2024 standardization of Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM), Module-Lattice-Based Digital Signature Algorithm (ML-DSA) and Stateless Hash-Based Digital Signature Algorithm (SLH-DSA). Beyond surveying these areas, we contribute three syntheses intended to make the material actionable: an analysis of compositional compatibility identifying where combining techniques creates conflicts rather than complementarity; a scenario-driven selection guide mapping deployment situations to technique combinations; and an examination of the gap between research prototypes and production systems spanning engineering complexity, key management, regulatory compliance and evidence from healthcare, finance and smart-city deployments. We close by identifying the open problems that will shape the next phase of research.

KEYWORDS: Privacy-preserving machine learning; homomorphic encryption; differential privacy; secure multi-party computation; federated learning; zero-knowledge proofs; searchable encryption; post-quantum cryptography; privacy budget; privacy auditing; large language models

1 Introduction

Machine learning systems increasingly operate on data whose disclosure would cause concrete harm. Hospitals train diagnostic models on patient images, banks build fraud detectors from transaction streams, and mobile keyboards learn language models from what users type. In each case the value of the model is inseparable from the sensitivity of the data that produced it. This tension has elevated privacy from a

Copyright © 2026 The Authors. Published by Tech Science Press. This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

compliance checkbox to a central engineering concern, and it has motivated a body of work that aims to extract the statistical value of data while provably limiting what an adversary can learn about any individual record. Collectively these techniques are referred to as privacy-preserving machine learning (PPML), and they draw on cryptography, statistics, and systems engineering in roughly equal measure [1,2].

The threats that PPML must address are not hypothetical. Membership inference attacks demonstrate that a model can betray whether a specific record was part of its training set, and modern formulations achieve high true-positive rates at low false-positive rates by reasoning carefully about the likelihood of observed model behavior under competing hypotheses [3,4]. Gradient-inversion attacks reconstruct training images pixel for pixel from the gradients shared during collaborative training [5,6]. These results establish that intuitive notions of privacy—"we only share aggregates," "we only share gradients," "the data never leaves the device"—are insufficient on their own. Rigorous guarantees require either cryptographic protection of the computation, formal perturbation of its outputs, or both.

Three broad families of techniques have emerged, and they organize the structure of this review: cryptographic approaches that protect the confidentiality of inputs and, through zero-knowledge proofs, the integrity of computation; perturbation-based differential privacy, which bounds what a released output reveals; and system- or storage-oriented approaches, which supply the architectural context in which the first two are deployed. Fig. 1 sets out the resulting taxonomy.

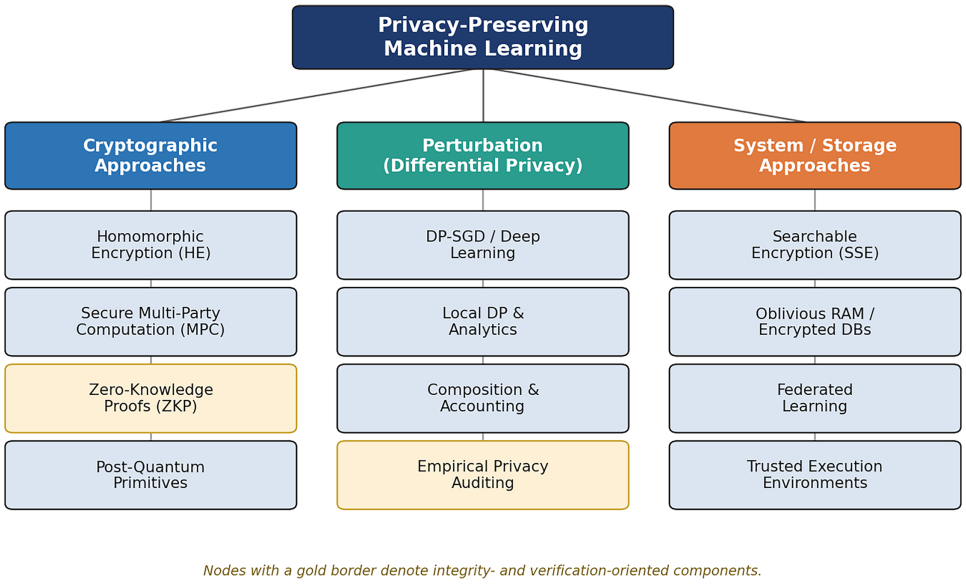


Figure 1: A taxonomy of privacy-preserving machine learning techniques, organized into three branches: cryptographic approaches (homomorphic encryption, secure multi-party computation, zero-knowledge proofs and post-quantum primitives), perturbation-based differential privacy (DP-SGD, local DP, composition accounting and empirical auditing), and system- or storage-oriented approaches (searchable encryption, oblivious RAM and encrypted databases, federated learning, and trusted execution environments). Branches are complementary rather than mutually exclusive: the confidentiality primitives of the left branch protect inputs, the perturbation mechanisms of the center branch protect released outputs, the integrity primitives (zero-knowledge proofs) protect correctness, and the right branch supplies the system context in which they are composed. Nodes with a gold border denote integrity- and verification-oriented components discussed in Sections 3.5, 5.4 and 7.5.

No single technique dominates. Homomorphic encryption offers strong confidentiality but imposes orders-of-magnitude computational overhead; differential privacy is computationally cheap but degrades

utility and provides only statistical, not cryptographic, protection; MPC achieves exact computation but consumes substantial communication; and federated learning reduces data movement but remains vulnerable to inference from the very updates it exchanges [7–9]. The art of building a privacy-preserving system lies in selecting and composing techniques to match a specific threat model, performance budget, and regulatory regime. The purpose of this review is to equip researchers with the conceptual map and quantitative anchors needed to make those choices.

We make five contributions. First, we provide a unified treatment of seven thematic areas that are usually surveyed in isolation—homomorphic encryption, differential privacy, privacy accounting, secure multi-party computation, federated learning, encrypted search, and post-quantum cryptography—drawing explicit connections among them and adding zero-knowledge proofs as the integrity counterpart to these confidentiality and privacy mechanisms [10,11]. Second, we collect quantitative results from primary sources—accuracy, latency, throughput, key sizes, and overhead factors—and present them in a form that supports cross-technique comparison while stating explicitly why such comparisons remain unreliable and what a unified benchmark would require (Section 13). Third, and going beyond the assertion of complementarity that surveys customarily make, we analyze compositional compatibility directly, identifying the pairings in which techniques conflict rather than combine—differential-privacy noise against homomorphic error growth, cryptographic communication against the bandwidth savings that motivate federation, and confidentiality against the visibility that robustness requires (Section 10). Fourth, we translate the survey into a decision procedure through a scenario-driven selection guide that maps deployment situations to technique combinations (Section 11). Fifth, we examine the gap between research prototypes and production systems, covering engineering complexity, key management, regulatory compliance, and documented deployment evidence from healthcare, finance, and smart-city settings (Section 12) [12,13]. The remainder of the paper is organized as follows. Section 2 establishes background, threat models, and notation. Sections 3 through 9 treat homomorphic encryption, differential privacy, privacy budget management, secure multi-party computation, federated learning, encrypted search and databases, and post-quantum cryptography respectively, each closing with a comparative perspective relating it to the others. Sections 10 to 13 present the compositional analysis, selection guide, deployment analysis, and benchmarking discussion. Section 14 synthesizes cross-cutting challenges and Section 15 concludes.

The scope of this review is deliberately broad but not exhaustive. We concentrate on the cryptographic and statistical mechanisms that provide formal privacy guarantees and on their specialization to machine learning, rather than on adjacent topics such as anonymization heuristics, access control, or hardware enclaves, which protect privacy through different and largely orthogonal means. We also focus on the protection of data and models rather than on fairness, robustness to adversarial examples, or explainability, which are distinct trustworthy-machine-learning concerns that interact with privacy but are governed by their own literatures. Where these adjacent areas bear directly on a privacy mechanism—as robustness does on federated learning, or as the approximate-decryption issue does on homomorphic encryption—we draw the connection explicitly [7,14].

A note on terminology is warranted because the same words carry different meanings across the communities this review spans. “Security” in the cryptographic sense means indistinguishability under a hardness assumption, whereas “privacy” in the differential-privacy sense means statistical indistinguishability of neighboring datasets; “leakage” is a precisely quantified side channel in searchable encryption but an informal notion of information disclosure elsewhere; and “threat model” ranges from the semi-honest-vs.-malicious dichotomy of multi-party computation to the auxiliary-information assumptions of inference attacks. We adopt each community’s usage within its section and flag the points of contact where a single system must satisfy definitions drawn from several traditions at once [1,8].

2 Background and Threat Models

A precise threat model is the prerequisite for any meaningful privacy claim. The techniques surveyed here differ less in the problems they solve than in the adversaries they assume, and conflating these assumptions is a common source of overstated guarantees. We therefore begin by fixing terminology.

2.1 Adversarial Assumptions

Cryptographic protocols are typically analyzed against one of two adversary classes. A semi-honest (or honest-but-curious) adversary follows the protocol faithfully but attempts to infer additional information from the messages it observes; a malicious (or active) adversary may deviate arbitrarily, sending malformed messages or aborting to extract information or corrupt the result. Most practical PPML systems target semi-honest security because it is dramatically cheaper, and they augment it with mechanisms—commitments, zero-knowledge proofs, or trusted hardware—only where active attacks are credible [9,15]. A second axis is the corruption threshold: protocols differ in how many of the participating parties may collude. Two-party and three-party honest-majority settings dominate the machine-learning literature because they admit highly efficient protocols, whereas dishonest-majority settings require heavier cryptographic machinery [16,17].

Statistical privacy, by contrast, is analyzed against a computationally unbounded adversary who observes the output of a mechanism and possibly auxiliary information about all but one record. Differential privacy guarantees that such an adversary cannot confidently determine whether any single record was present, regardless of side knowledge [18,19]. This is a fundamentally different and in some respects stronger guarantee than cryptographic confidentiality, because it bounds what can be inferred from the legitimate output itself rather than only protecting the intermediate computation.

A further distinction concerns the strength of the guarantee itself. Cryptographic protocols offer computational security: they are secure against any polynomial-time adversary under a hardness assumption such as the difficulty of the learning-with-errors problem. A smaller set of techniques—notably information-theoretic secret sharing and certain secure-aggregation masks—offer unconditional security that holds even against an unbounded adversary, but only under stronger structural assumptions such as an honest majority. Differential privacy occupies a third category: its guarantee is statistical and unconditional, independent of any computational assumption, but it protects the output distribution rather than the inputs [18,19]. Understanding which category a system inhabits is essential, because a protocol that is computationally secure under a classical assumption may be retroactively broken by a quantum adversary, a concern that motivates the entire post-quantum agenda of Section 9 [20].

2.2 What is Being Protected

It is useful to distinguish the asset under protection. Input privacy concerns the confidentiality of the data fed into a computation—the patient image submitted for classification. Output privacy concerns what the released result reveals about individuals—whether a published model or aggregate statistic leaks training records. Cryptographic techniques such as HE and MPC primarily provide input privacy: they guarantee that parties learn nothing beyond the agreed output, but they say nothing about whether that output is itself disclosive. Differential privacy provides output privacy: it bounds the inferential power of the result but does not hide the inputs from a party that legitimately holds them. Because the two are orthogonal, robust systems often combine them, using MPC or HE to compute a function and DP to sanitize its output [21,22].

A third asset, distinct from inputs and outputs, is the model itself. In many commercial settings the trained model is valuable intellectual property, and the client who submits data for inference must not be

able to extract the model's weights, while the server hosting the model must not learn the client's data. This two-sided confidentiality requirement is exactly what hybrid inference protocols target, and it explains why so much effort is devoted to protocols in which neither party's secret is revealed even as a joint computation proceeds [16,23]. Model-extraction and model-inversion attacks, which attempt to reconstruct weights or training data from query access, show that protecting the model and protecting the data are intertwined rather than independent goals [3,4].

2.3 The Cost Dimensions

Every privacy technique extracts a price along one or more axes: computational latency, communication volume, memory footprint, and utility loss. Homomorphic encryption converts a problem of confidentiality into a problem of computational throughput, since arithmetic on ciphertexts is far slower than on plaintexts and ciphertext sizes inflate memory and bandwidth. Multi-party computation converts confidentiality into communication, since parties must exchange large volumes of correlated randomness and intermediate shares. Differential privacy converts confidentiality into utility loss, since noise degrades accuracy. Recognizing which resource a technique consumes is essential for matching it to a deployment whose bottleneck—compute, network, or accuracy—is known. The sections that follow quantify these costs for representative systems.

Quantifying utility loss requires care because the appropriate metric is task-dependent. For classification, the natural measure is the gap in test accuracy between the private and non-private model; for regression and generative tasks, it is the increase in error or the degradation in sample quality; and for analytics, it is the relative error of released statistics. A recurring methodological pitfall is to report a favorable privacy budget while measuring utility on a task or dataset for which noise happens to be benign, which can make a mechanism appear more practical than it is [8]. Throughout this review we report the metric used by each primary source and avoid normalizing across incompatible measures.

2.4 Notation, Terminology and Abbreviations

Because this review spans several research communities, Table 1 lists the abbreviations used throughout, and Table 2 fixes the symbols that recur in later sections.

Table 1: Abbreviations used in this review, listed alphabetically. Each abbreviation is also expanded at its first occurrence in the abstract and in the main text.

Abbreviation	Definition
ABY3	Three-party mixed-protocol framework for privacy-preserving machine learning
AES	Advanced Encryption Standard
ASIC	Application-Specific Integrated Circuit
BatchCrypt	Batched homomorphic encryption scheme for cross-silo federated learning
BERT	Bidirectional Encoder Representations from Transformers
BFV	Brakerski/Fan–Vercauteren homomorphic encryption scheme
BGV	Brakerski–Gentry–Vaikuntanathan homomorphic encryption scheme
BOLT	Privacy-preserving transformer inference framework
BumbleBee	Secure two-party inference framework for large transformers
CHET	Compiler for homomorphic neural-network inference
CIFAR	Canadian Institute for Advanced Research image dataset
CKKS	Cheon–Kim–Kim–Song approximate homomorphic encryption scheme
CNN	Convolutional Neural Network

(Continued)

Table 1 (continued)

Abbreviation	Definition
CoeffToSlot	Coefficient-to-slot linear transformation used in CKKS bootstrapping
CRYSTALS-Dilithium	Lattice-based digital signature scheme standardized as ML-DSA
CRYSTALS-Kyber	Lattice-based key-encapsulation mechanism standardized as ML-KEM
CryptDB	Encrypted relational database system supporting SQL query processing
CrypTen	Secure multi-party computation framework with a tensor programming interface
CryptoNets	Early framework for neural-network inference on homomorphically encrypted data
CUDA	Compute Unified Device Architecture (NVIDIA GPU programming platform)
DM/CGGI	Ducas–Micciancio and Chillotti–Gama–Georgieva–Izabachène bootstrapping approach
DP	Differential Privacy
DP-SGD	Differentially Private Stochastic Gradient Descent
EVA	Encrypted Vector Arithmetic language and compiler
FedML	Open federated-learning research and deployment library
FHE	Fully Homomorphic Encryption
FHEW	Fastest Homomorphic Encryption in the West bootstrapping scheme
FIPS	Federal Information Processing Standard
FN-DSA	FFT over NTRU-Lattice Digital Signature Algorithm (derived from Falcon)
FPGA	Field-Programmable Gate Array
GAZELLE	Low-latency framework for secure neural-network inference
GELU	Gaussian Error Linear Unit activation function
GSW	Gentry–Sahai–Waters homomorphic encryption construction
HE	Homomorphic Encryption
HEaaN	Homomorphic Encryption for Arithmetic of Approximate Numbers library
HElib	Homomorphic Encryption library implementing BGV and CKKS
HEonGPU	GPU-based fully homomorphic encryption library
HQC	Hamming Quasi-Cyclic code-based key-encapsulation mechanism
LLM	Large Language Model
LLaMA-7B	Seven-billion-parameter open large language model
LoRA	Low-Rank Adaptation
LUT	Look-Up Table
META-BTS	Bootstrapping procedure extending CKKS precision beyond the single-iteration limit
ML-DSA	Module-Lattice-Based Digital Signature Algorithm (FIPS 204)
ML-KEM	Module-Lattice-Based Key-Encapsulation Mechanism (FIPS 203)
MNIST	Modified National Institute of Standards and Technology handwritten-digit dataset
MPC	Secure Multi-Party Computation
NEXUS	Non-interactive secure transformer inference framework
NIST	National Institute of Standards and Technology
NTT	Number-Theoretic Transform
OpenFHE	Open-source fully homomorphic encryption library
ORAM	Oblivious Random-Access Machine
PaCo	Partial coefficient-to-slot CKKS bootstrapping procedure
PIGEON	High-throughput framework for private neural-network inference under MPC
PPML	Privacy-Preserving Machine Learning
PUMA	Three-party secure inference framework for large language models
RAM	Random-Access Memory
ReLU	Rectified Linear Unit activation function
ResNet	Residual Network convolutional architecture
RLWE	Ring Learning with Errors

(Continued)

Table 1 (continued)

Abbreviation	Definition
RNS-CKKS	Residue Number System variant of the CKKS scheme
RSA	Rivest–Shamir–Adleman public-key cryptosystem
SEAL	Simple Encrypted Arithmetic Library
SecretFlow-SPU	Secure Processing Unit compiler and runtime for privacy-preserving machine learning
SHIP	Shallow and Highly Parallelizable CKKS bootstrapping algorithm
SIMD	Single Instruction, Multiple Data
SLH-DSA	Stateless Hash-Based Digital Signature Algorithm (FIPS 205)
SlotToCoeff	Slot-to-coefficient linear transformation used in CKKS bootstrapping
SPDZ	Dishonest-majority multi-party computation protocol family
SPHINCS+	Stateless hash-based signature framework standardized as SLH-DSA
SQL	Structured Query Language
SSE	Searchable Symmetric Encryption
TFHE	Fast Fully Homomorphic Encryption over the Torus
THOR	Secure transformer inference framework based on homomorphic encryption
ZKP	Zero-Knowledge Proof
zkCNN	Zero-knowledge proof system for convolutional neural-network inference

Note: Scheme, library and system names are included where they are commonly written in abbreviated or acronym form.

Table 2: Notation used throughout this review, with the section in which each symbol is principally used. Symbols are stated once here so that formulas in later sections need not redefine them.

Symbol	Meaning	Principal Use
ε	Privacy budget; multiplicative bound on the change in output probability between neighboring datasets. Smaller is more private.	Sections 4 and 5
δ	Additive failure probability in (ε, δ)-differential privacy; conventionally $\delta \ll 1/n$.	Sections 4 and 5
n	Number of records in the dataset, or number of participating clients in federated learning.	Sections 4 and 7
Δf	Sensitivity of a query f: the largest change in its output when one record changes (L1 or L2 norm).	Section 4
σ	Standard deviation of the Gaussian noise added by a mechanism; the noise multiplier is $\sigma/\Delta f$.	Sections 4 and 5
C	Per-example gradient clipping norm in DP-SGD, which bounds sensitivity.	Section 4.2
q	Sampling rate (expected batch size divided by dataset size), which governs privacy amplification.	Sections 4.2 and 5.2
T	Number of composed mechanism invocations, e.g., training iterations.	Section 5
α	Order of the Rényi divergence in Rényi differential privacy.	Section 5.3

(Continued)

Table 2 (continued)

Symbol	Meaning	Principal Use
N	Ring dimension of the cyclotomic ring underlying an RLWE-based scheme; a power of two, typically 2^{13}–2^{17}.	Section 3
Q	Ciphertext modulus; the modulus chain length determines the available multiplicative depth.	Section 3
L	Multiplicative depth consumed before bootstrapping is required.	Sections 3.1 and 3.2
m	Number of parties in a multi-party computation protocol.	Section 6
t	Corruption threshold: the number of parties an adversary may control.	Sections 2.1 and 6

Note: Where a section introduces additional local notation, it is defined at the point of use.

Because this review spans communities that use overlapping vocabulary differently, we fix notation and terminology once here and use it consistently throughout. Table 2 collects the symbols that recur in the sections that follow. Two conventions deserve emphasis. First, we reserve "privacy" for guarantees about what an adversary can infer about an individual record, and "confidentiality" for guarantees that data or intermediate values are hidden from a party; a protocol can provide confidentiality without privacy, and differential privacy provides privacy without confidentiality. Second, we use "leakage" in the precise sense established in the searchable-encryption literature, namely the well-defined function of the data and query sequence that a scheme permits the adversary to observe, and we avoid the looser informal usage found elsewhere [24,25].

3 Homomorphic Encryption for Machine Learning

Homomorphic encryption (HE) enables computation on encrypted data: a party holding only ciphertexts can evaluate a function and obtain an encryption of the result, which only the secret-key holder can decrypt. The idea dates to the privacy homomorphisms of Rivest et al. [26], but it remained largely theoretical until lattice-based constructions made it practical. For machine learning, HE is attractive because it offers strong, non-interactive input privacy under a single-server model: a client encrypts its data, the server computes the model, and the client decrypts the prediction, with the server learning nothing [7].

3.1 Fully Homomorphic Encryption Schemes

Four schemes dominate contemporary practice, all built on the hardness of variants of the learning-with-errors (LWE) problem. The Brakerski–Gentry–Vaikuntanathan (BGV) scheme introduced leveled fully homomorphic encryption, supporting a bounded number of multiplications without the expensive bootstrapping step by managing noise growth through modulus switching [27]. The Brakerski/Fan–Vercauteren (BFV) scheme provides similar exact integer arithmetic with a scale-invariant noise structure [28]. A fifth construction, Gentry–Sahai–Waters (GSW), is not deployed directly for machine-learning workloads but established homomorphic encryption directly from LWE with a conceptually simple structure that underpins the FHEW/TFHE line and much later analysis [29]; because it is not used as a standalone scheme for encrypted inference, it is discussed here but omitted from the four-scheme comparison in Table 3. For machine learning the most important scheme is Cheon–Kim–Kim–Song (CKKS), which supports

approximate arithmetic over real and complex numbers and includes a rescaling operation that makes it natural for the fixed-point computations of neural networks [30]. For Boolean and small-integer workloads, the TFHE scheme offers very fast gate-by-gate bootstrapping, reducing the bootstrapping time of a single gate to roughly thirteen milliseconds on a single core while shrinking the bootstrapping key from gigabytes to sixteen megabytes [31]. Table 3 compares the schemes along the dimensions that most affect their suitability for machine learning.

Table 3: Comparison of the principal fully homomorphic encryption schemes used in privacy-preserving machine learning. All four rest on the hardness of (ring/module) learning-with-errors and are therefore post-quantum candidates; all are instantiated at a target security level of 128 bits in the works surveyed. "Plaintext domain" is the algebraic structure the scheme natively encrypts; "Arithmetic" states whether results are exact or approximate and whether many plaintext slots are processed per operation (SIMD packing, typically 2^{12}–2^{16} slots at ring dimension $N = 2^{13}$–2^{17}); "Bootstrapping style" names the dominant noise-refreshing strategy, for which several alternatives now exist (Section 3.2).

Scheme	Plaintext Domain	Arithmetic	Bootstrapping Style	Primary ML Use
BGV	Integers (modular)	Exact, SIMD-batched	Leveled; infrequent	Polynomial evaluation, statistics
BFV	Integers (modular)	Exact, SIMD-batched	Leveled; infrequent	Integer model inference
CKKS	Real/complex (approx.)	Approximate, SIMD-batched	Approximate modular reduction	Deep neural-network inference
TFHE	Bits/small integers	Boolean/LUT	Fast gate-by-gate	Non-linear functions, decision trees

Note: All schemes support single-instruction-multiple-data (SIMD) packing of many plaintext slots into one ciphertext; effective throughput depends strongly on how well a workload maps onto this packing.

Across all schemes, security parameters are chosen so that the underlying lattice problem resists the best known classical and quantum attacks, which is also why HE is intrinsically a post-quantum technology, a point we return to in Section 9 [7].

The security and performance of all four schemes flow from the structure of the ring learning-with-errors problem. Plaintexts are encoded into polynomials in a cyclotomic ring, and a ciphertext is a pair of such polynomials whose structure hides the message under a small "error" term sampled from a discrete Gaussian. Homomorphic addition adds ciphertext polynomials and roughly adds their errors; homomorphic multiplication is more delicate, multiplying ciphertexts in a way that multiplies their errors and expands the ciphertext, necessitating a relinearization step that uses an evaluation key to restore the compact two-polynomial form. The steady accumulation of error across operations is the fundamental obstacle that bootstrapping exists to overcome [27,28].

A crucial efficiency mechanism shared by BGV, BFV, and CKKS is single-instruction-multiple-data packing: the Chinese-remainder structure of the cyclotomic ring lets a single ciphertext encode thousands of independent plaintext slots, so that one homomorphic operation processes all slots in parallel. For machine learning this amortization is decisive, because a model applied to a large batch of inputs, or a matrix multiplication expressed through rotations across slots, can exploit the full width of the packing. CKKS additionally treats the encoding as fixed-point, with a rescaling operation that divides out an accumulated scaling factor after each multiplication, mirroring the way fixed-point arithmetic controls magnitude in

quantized neural networks [30]. This correspondence is why CKKS, rather than the exact-integer schemes, has become the default substrate for encrypted deep learning.

The approximate-decryption subtlety noted above has a precise cryptographic formulation: because a CKKS decryption returns a value that still carries the computation's error term, releasing that value to an adversary who supplied the inputs can reveal information about the secret key, so that the standard notion of indistinguishability under chosen-plaintext attack does not by itself suffice in the approximate setting [32]. The accepted remedy is to add a small amount of noise to the decrypted result before release, calibrated so that the residual error is statistically independent of the secret. This is precisely a differential-privacy-style output perturbation applied for a cryptographic purpose, and it is the clearest single illustration of the compositional theme developed in Section 10 [18].

3.2 Bootstrapping

Every homomorphic operation increases the noise embedded in a ciphertext, and once that noise exceeds a threshold the ciphertext can no longer be decrypted correctly. Bootstrapping—homomorphically evaluating the decryption circuit to produce a fresh, low-noise ciphertext—is what elevates a leveled scheme to a fully homomorphic one capable of unbounded computation. It is also the dominant performance cost, and a large body of recent work attacks it. A systematization of bootstrapping clarifies the trade-off space: CKKS bootstrapping offers the highest throughput for vectors of real numbers, the regime relevant to machine learning, whereas the FHEW/TFHE family achieves the lowest latency for individual small-integer look-up tables [33]. META-BTS extended the achievable precision of CKKS bootstrapping beyond the previous single-iteration limit [34], and subsequent work has reduced its asymptotic and concrete cost: a partial CoeffToSlot decomposition reduces the computational complexity to a logarithmic function of the number of slots [35], while level-conserving rescaling saves a modulus level and improves throughput by roughly twenty to thirty-five percent at comparable parameters [36]. These advances, layered on hardware acceleration discussed below, have brought bootstrapped deep-network inference from the realm of the impossible into that of the merely expensive.

The cost structure of bootstrapping differs sharply between scheme families, and this difference dictates which scheme suits which workload. In the CKKS setting, bootstrapping homomorphically evaluates an approximation to the modular-reduction function using a composition of linear transformations (the so-called CoeffToSlot and SlotToCoeff steps) and a polynomial approximation of a sine or related function; the linear transformations dominate the cost, which is why reducing their complexity—as in the partial-decomposition approach—directly improves throughput [35,37]. In the TFHE setting, by contrast, bootstrapping is performed after every gate using a programmable look-up table realized through blind rotation of an accumulator, which is what gives TFHE its very low per-gate latency and its natural support for arbitrary non-linear functions [31].

These complementary cost profiles have given rise to scheme switching, in which a computation is performed in CKKS for its high-throughput arithmetic on real vectors and temporarily converted to a TFHE-style representation to evaluate a non-linear function such as a comparison or activation, then converted back. Library support for such switching, together with the hundredfold hardware accelerations discussed in Section 3.4, has been the practical enabler of bootstrapped deep-network inference [33,38]. The research frontier is the amortized cost per useful operation: recent CKKS bootstrapping work reports amortized costs in the microseconds-per-slot range for suitable parameter sets, a regime that was unthinkable when bootstrapping was introduced [34,36].

The description above characterizes CKKS bootstrapping by its classical route, the homomorphic evaluation of a polynomial approximation to modular reduction [37], but this is now only one of several

established styles, and a reader surveying the last two years of the literature would find the picture considerably richer. A first alternative dispenses with polynomial approximation of the modular-reduction function altogether by embedding the additive group modulo the base modulus into the complex roots of unity, which CKKS can evaluate natively. Because this removes the deep polynomial evaluation that dominates classical bootstrapping depth, it permits smaller ring dimensions and lower latency: the SHIP algorithm achieves a very shallow multiplicative depth and, through a combination of reduced ring size and high parallelizability, low end-to-end latency [39], and a closely related roots-of-unity construction reports up to a fivefold latency reduction using an open-source library implementation [40]. The PaCo procedure reformulates the decryption circuit in terms of blind rotations and modular additions realized through the circle group in the complex plane, achieving complexity logarithmic in the number of slots [35].

A second and conceptually distinct development is functional bootstrapping, in which the refresh operation simultaneously evaluates an arbitrary function expressed as a look-up table, rather than merely restoring noise budget. This capability was long the preserve of the FHEW/TFHE family, whose programmable bootstrapping evaluates a look-up table on every gate but must bootstrap each encrypted value separately. Recent work has brought general functional bootstrapping to CKKS using a toolkit based on trigonometric Hermite interpolation, retaining the amortization advantage of CKKS packing: for eight-bit look-up tables the reported amortized time is 0.72 ms per value, three orders of magnitude faster than the DM/CGGI approach and 6.8 times faster than a more restrictive amortized functional bootstrapping method built on BFV [41]. Related constructions target the discrete end of the spectrum, bootstrapping ciphertexts that encrypt bits or small integers within the CKKS framework and thereby narrowing the historical division of labor between word-wise and bit-wise schemes [42,43].

These developments matter directly for machine learning because they change which architectural choices are expensive. If an arbitrary activation function can be evaluated as a look-up table during a refresh that must occur anyway, the long-standing pressure to replace ReLU and GELU with low-degree polynomial surrogates is substantially reduced, and the accuracy loss that such surrogates introduce can be avoided. The complementary route is scheme switching, in which a computation proceeds in CKKS for throughput and converts temporarily to a bit-wise representation to evaluate a comparison or activation, as demonstrated by bridging frameworks that combine polynomial and non-polynomial evaluation [44]; this approach is now supported natively by unified libraries [33,38]. Table 3 summarizes the schemes; the bootstrapping styles now available within CKKS alone comprise classical approximate modular reduction, roots-of-unity and blind-rotation constructions, and general functional bootstrapping, each occupying a different point in the depth, precision and generality trade-off.

3.3 Encrypted Neural-Network and Transformer Inference

The trajectory of encrypted inference traces the maturation of the field. CryptoNets first demonstrated convolutional-network inference over encrypted data, reporting roughly ninety-nine percent accuracy on the MNIST digit task at a throughput near fifty-one thousand predictions per hour, achieved by packing thousands of independent inputs into the SIMD slots of a single ciphertext [45]. The price was a per-batch latency measured in minutes and a restriction to shallow networks with polynomial activations. Optimizing compilers such as CHET and EVA automated the error-prone tasks of rescaling, relinearization, and parameter selection, making it feasible to target CKKS from higher-level descriptions [46,47].

The decisive step toward deep models came with bootstrapped residual networks. A full implementation of ResNet-20 under the residue-number-system variant of CKKS reported classification accuracy of 92.43 percent on CIFAR-10, within a fraction of a percent of the plaintext model, with the encrypted output matching the plaintext model on more than ninety-eight percent of inputs; the cost was on the order of

hours per image on a dual-socket server [48]. Multiplexed parallel convolutions and an imaginary-removing bootstrapping procedure then reduced inference latency by a factor of roughly 4.7 and amortized runtime by more than two orders of magnitude, and enabled the first implementation of ResNet-110 over the residue-number-system CKKS scheme [49]. Specialized optimizations for convolutional inference under CKKS have continued to narrow the gap to plaintext performance [50]. Fig. 2 places representative latencies on a common logarithmic axis, with the caveat that the underlying experiments differ in hardware, model, and dataset and are therefore only loosely comparable.

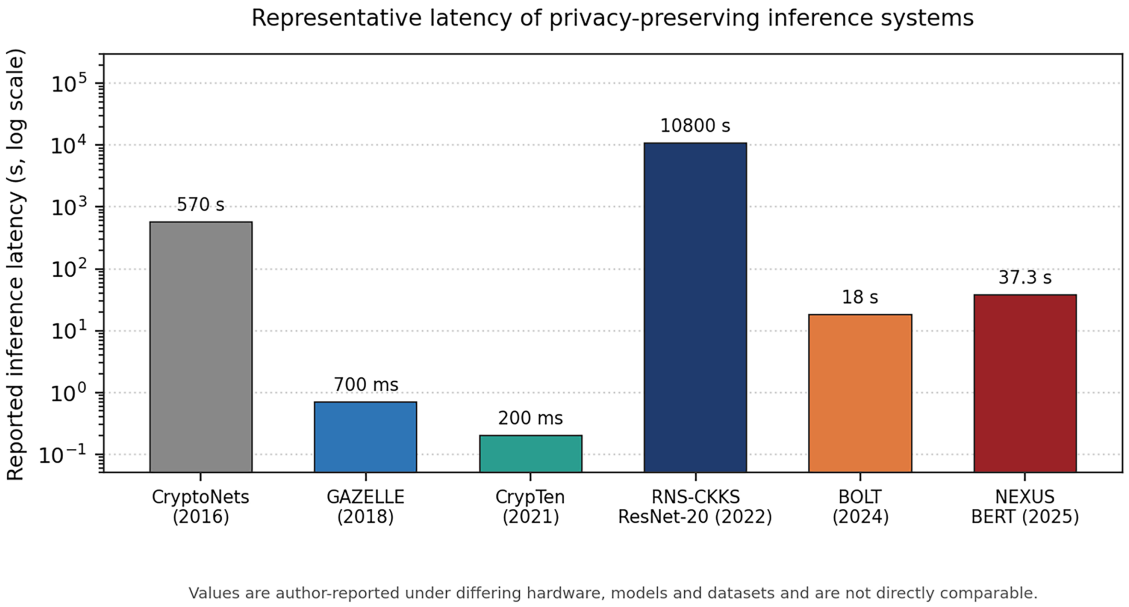


Figure 2: Representative reported inference latencies for privacy-preserving inference systems, on a logarithmic scale spanning roughly five orders of magnitude. Bars show wall-clock latency in seconds as reported by the primary source for that system: CryptoNets (shallow CNN, MNIST, batch-amortized), GAZELLE and CrypTen (small CNNs, interactive protocols), RNS-CKKS ResNet-20 (CIFAR-10, bootstrapped, per image), BOLT and NEXUS (BERT-scale transformers). Because the underlying experiments differ in hardware platform, model architecture, dataset, batch size, threat model and security parameter, the figure conveys orders of magnitude only and is not a controlled benchmark; Section 13 discusses why such comparisons remain unreliable and what a unified benchmark would require.

The current frontier is secure inference of transformer and large-language-model architectures, whose softmax, GELU, and layer-normalization operations are far less amenable to homomorphic evaluation than the piecewise-linear activations of convolutional networks. Hybrid protocols that combine HE for linear layers with interactive secret sharing for non-linearities have advanced rapidly: BOLT introduced compact packing and baby-step giant-step optimizations to reduce homomorphic rotations, achieving accurate and efficient transformer inference [23], and BumbleBee compressed the resulting ciphertexts to lower communication for large models [51]. Purely non-interactive approaches have also appeared: NEXUS makes secure transformer inference non-interactive and reports several-hundred-fold bandwidth reductions relative to interactive baselines, running BERT-scale inference in tens of seconds [52], while THOR provides a non-interactive homomorphic framework using diagonal-major matrix-multiplication encodings [53]. On the secret-sharing side, PUMA evaluates LLaMA-7B in minutes by designing high-quality polynomial approximations for expensive non-linear functions [54], and co-design of the model itself—replacing attention non-linearities with encryption-friendly alternatives—has been shown to reduce cost substantially while preserving accuracy [55]. Table 4 summarizes representative encrypted-inference systems.

Table 4: Representative privacy-preserving inference systems and their reported characteristics. "Protocol" distinguishes HE-only designs (non-interactive, server-side computation on ciphertexts), MPC/secret-sharing designs (interactive, communication-bound), and hybrid designs that evaluate linear layers homomorphically and non-linearities interactively. The results are reported by the respective authors under each paper's own hardware, dataset and security parameters and are not normalized across works; throughput-oriented systems optimize amortized cost over large batches whereas latency-oriented systems optimize single-query response, so the two families are not directly comparable (Section 13).

System	Protocol	Target Model	Reported Result
CryptoNets	HE (leveled)	Shallow CNN (MNIST)	~99% accuracy; high-throughput SIMD batching
RNS-CKKS ResNet	HE (bootstrapped)	ResNet-20 (CIFAR-10)	92.43% accuracy; hours per image
Multiplexed conv.	HE (bootstrapped)	ResNet-20/110	~4.7× lower latency; first ResNet-110
BOLT	Hybrid HE+MPC	BERT transformer	Accurate, efficient transformer inference
NEXUS	HE (non-interactive)	BERT transformer	Tens of seconds; large bandwidth reduction
PUMA	MPC (3-party)	LLaMA-7B	Inference in minutes via polynomial approx.

Note: Cross-system comparison is approximate: throughput-oriented systems optimize amortized cost over large batches, whereas latency-oriented systems optimize single-query response.

Two design constraints shape every encrypted-inference system. First, homomorphic schemes evaluate polynomials efficiently but cannot directly compute the non-polynomial activations—ReLU, sigmoid, softmax—that modern networks rely on; these must be replaced by polynomial approximations or evaluated through interaction or look-up tables, and the quality of these approximations largely determines both accuracy and cost. Second, the multiplicative depth of the network determines how much noise budget is consumed and therefore how frequently bootstrapping is required, so architectures with lower depth are dramatically cheaper to evaluate. Much of the systems work in this area can be read as managing these two constraints: CryptoNets used low-degree polynomial activations and a shallow network to avoid bootstrapping entirely [45], whereas the bootstrapped ResNet implementations accepted the cost of frequent noise refreshing in exchange for the ability to evaluate deep, accurate models [48,49].

The transition to transformers sharpens these constraints because attention couples every position to every other through a softmax, and layer normalization introduces division and square-root operations that are expensive under encryption. The systems surveyed adopt different strategies: hybrid protocols evaluate the linear projections homomorphically and delegate softmax and GELU to interactive secret-sharing subprotocols, accepting communication cost in exchange for accuracy [23,51]; non-interactive systems push the entire computation into homomorphic evaluation using carefully designed polynomial approximations and packing schemes, trading some accuracy and substantial offline computation for the elimination of round-trip latency [52,53]; and co-design approaches modify the model architecture itself, for example replacing softmax with a polynomial-friendly normalization, so that the resulting network is intrinsically cheaper to evaluate under encryption [55]. The reported figures—BERT-scale inference in tens of seconds and LLaMA-7B inference in minutes—mark dramatic progress, but they remain far from the millisecond latencies of plaintext serving, which is why this is among the most active areas of the field [52,54].

It is worth dwelling on why throughput-oriented and latency-oriented systems are difficult to compare, because the distinction recurs throughout privacy-preserving machine learning. A throughput-oriented system such as CryptoNets packs many independent queries into the slots of a single ciphertext and reports an amortized cost per query that is low precisely because the fixed cost of the homomorphic operations is spread across thousands of inputs; its single-query latency, however, may be minutes [45]. A latency-oriented system optimizes the response time for one query, which is the metric that matters for interactive inference, but cannot amortize fixed costs and so reports a higher per-query cost in batch settings. A bar chart of

latencies, such as Fig. 2, therefore conflates two genuinely different operating points, and the practitioner must ask which regime a reported number describes before drawing conclusions [48,52]. This review reports the regime alongside the number wherever the primary source makes it clear.

3.4 Libraries and Hardware Acceleration

Practical progress has depended on mature open-source libraries. OpenFHE consolidates a decade of implementation experience, supporting BGV, BFV, CKKS, and the FHEW/TFHE family with bootstrapping and scheme switching behind a hardware-abstraction layer [38]. Performance, however, remains gated by raw arithmetic throughput, which has motivated extensive work on acceleration. Memory-centric optimization on graphics processors yielded more than a hundredfold speedup in amortized bootstrapping [56], and unified CUDA libraries now provide word-wise homomorphic arithmetic across schemes on commodity GPUs [57]. General-purpose secure-computation frameworks such as CrypTen expose a familiar tensor programming model while executing the underlying cryptographic protocol, lowering the barrier to entry for machine-learning practitioners [58]. Taken together, software maturity and hardware acceleration have shifted the bottleneck for many workloads from feasibility to cost, which is precisely the regime in which the bootstrapping and protocol optimizations of the preceding subsections deliver their value.

The library ecosystem extends well beyond any single implementation. Microsoft SEAL popularized BFV and CKKS with a clean programming interface; HElib pioneered many BGV optimizations including efficient bootstrapping; HEaaN provided the reference CKKS implementation; Lattigo brought a pure-Go implementation with native support for multiparty and threshold variants; and Concrete provided a Rust implementation of TFHE oriented toward usability. OpenFHE was conceived as a successor that unifies these lineages under a common abstraction with a hardware-acceleration interface, which positions it well for the accelerator era [7,38].

Hardware acceleration has progressed along two tracks. On commodity graphics processors, memory-centric optimization addresses the fact that homomorphic operations are bandwidth-bound rather than compute-bound, restructuring data movement to achieve more than a hundredfold speedup in amortized bootstrapping [56], and unified CUDA libraries now expose word-wise homomorphic arithmetic across schemes [57]. On the custom-silicon track, a series of application-specific accelerators has targeted the number-theoretic transform and the large on-chip memory that bootstrapping demands, projecting further order-of-magnitude gains. The practical implication for machine-learning researchers is that the cost of encrypted inference is now a moving target set by hardware as much as by cryptography, and system designs should anticipate continued acceleration rather than treating today's latencies as fixed [50,58].

A practical concern that cuts across libraries is parameter selection, which couples security, correctness, and performance in a way that is easy to get wrong. The ring dimension, the ciphertext modulus chain, and the error distribution must jointly satisfy a target security level—typically 128 bits against the best known lattice attacks—while leaving enough noise budget for the intended computation. Misconfiguration can silently reduce security or cause decryption failures, which is why the optimizing compilers of Section 3.3 and the parameter advisories embedded in modern libraries are not mere conveniences but correctness-critical infrastructure [38,46,47]. For machine-learning researchers without a cryptography background, this automation is often the difference between a usable system and an insecure one.

The account of acceleration above is necessarily selective, and readers who need the complete picture should consult the two systematizations that now cover this ground. A dedicated systematization of homomorphic-encryption accelerators reviews fourteen designs spanning GPU, FPGA and ASIC targets and adds testbed measurements placing the open-source ones on a common footing [59], while a systematization of private deep-network inference devotes a full section to hardware acceleration organized by platform,

together with a comparative table of accelerator designs reporting word size, on-chip memory, design focus, frequency, power and area [60]. We summarize their principal findings here rather than reproduce a literature too large to survey in a section.

A single architectural fact organizes the whole area: homomorphic arithmetic is memory-bound rather than compute-bound, so the binding constraint is on-chip capacity and the scheduling of data movement rather than arithmetic throughput [59,60]. The number-theoretic transform dominates the instruction mix, and the key-switching operation that follows every rotation and relinearization is awkward to parallelize because its two components pull in opposite directions: the transform needs all coefficients of a single polynomial and therefore suits limb-level parallelism, whereas basis conversion parallelizes across coefficients but requires data from multiple limbs. Accelerator designs are largely distinguished by how they resolve that tension [60].

The platforms occupy distinct positions. CPU acceleration is the most limited, resting on vector instruction sets to vectorize modular arithmetic and the transform; gains are real on a single thread but degrade as thread count rises, and memory bandwidth caps the achievable speedup. GPUs offer both massive parallelism and higher memory bandwidth, which suits memory-bound kernels, at the cost of limited architectural customization. ASICs deliver the highest performance through wide vector units and very large scratchpads: the first programmable design established that complete homomorphic programs could be accelerated rather than individual kernels, a later design reduced key-switching auxiliary data from gigabytes to tens of megabytes, and subsequent designs targeted bootstrapping specifically, one of them eliminating the large majority of off-chip reads during the procedure. Because such designs become large monolithic chips that are costly to fabricate, the most recent work moves to cluster- and chiplet-based organizations for scalability. FPGAs occupy a middle ground, offering customizable datapaths for modular arithmetic at far lower engineering cost than ASICs, and high-bandwidth-memory parts have largely closed their earlier bandwidth disadvantage; designs range from early key-switching and transform engines without bootstrapping support to complete pipelines evaluated on multi-FPGA configurations. A fifth line attacks the memory wall directly through near-memory and processing-in-memory designs, although work on the one commercially available processing-in-memory platform has not yet demonstrated 128-bit security together with end-to-end timing results [60].

Two cautions from that literature deserve emphasis because they bear on the benchmarking argument of Section 13. First, accelerator evaluations have overwhelmingly used small convolutional models, typically ResNet-20 on CIFAR-10, which leaves their behavior on transformer- and language-model workloads largely unestablished. Second, among open-source libraries only OpenFHE currently provides internal parallelism for CKKS, so reported multi-threaded results are not straightforwardly portable across implementations [38,60].

The GPU library ecosystem has broadened well beyond the single implementation cited in the previous version of this review. Alongside Phantom [57], HEonGPU executes all homomorphic operations on the device and uses a multi-stream architecture that overlaps independent tasks while avoiding repeated host-to-device transfers, supporting BFV and CKKS and subsequently extended to support TFHE [61]. Liberate.FHE takes a different design position, implementing CKKS in pure Python and CUDA with native multi-GPU support and an explicit goal of being modifiable by non-cryptographers, which bears on the adoption barrier discussed in Section 12.1 [62]; it is the library underlying several recent private transformer implementations [53,60]. Earlier GPU work by the same group as HEonGPU accelerated BFV with optimized transform kernels and integrated with an established CPU library, illustrating the common pattern of GPU backends attaching to rather than replacing CPU libraries [63]. Acceleration is not confined to homomorphic encryption: on the multi-party side, oblivious-transfer extension is the latency bottleneck

of most secure-inference frameworks, and near-memory designs targeting it report substantial end-to-end gains [9,64]. Table 5 summarizes the landscape.

Table 5: Software libraries and hardware platforms for homomorphic encryption and secure computation. "Schemes" lists support at the time of writing and "bootstrapping" indicates whether noise refreshing is available in the implementation, which determines whether unbounded-depth circuits can be evaluated. Capabilities evolve rapidly and readers should consult current documentation. The final four rows aggregate hardware platform families rather than individual designs; the systematizations cited in Section 3.4 provide design-by-design comparisons including word size, on-chip memory, frequency, power and area.

Platform	Type	Schemes	Bootstrapping	Distinguishing Characteristic
SEAL	CPU library	BFV, CKKS	No	Widely adopted reference implementation
HElib	CPU library	BGV, CKKS	Yes	Pioneered many BGV optimizations
OpenFHE	CPU library	BGV, BFV, CKKS, FHEW/TFHE	Yes	Unified successor; only open-source library with internal CKKS parallelism
Lattigo	CPU library (Go)	BFV, CKKS	Yes	Native multiparty and threshold variants
Concrete	CPU library (Rust)	TFHE	Yes	Programmable bootstrapping; usability focus
HEaaN	CPU/GPU library	CKKS	Yes	Not open-source; widely used in reported GPU results
Phantom	GPU (CUDA)	BFV, BGV, CKKS	Yes	Word-wise arithmetic across schemes
HEonGPU	GPU (CUDA)	BFV, CKKS, TFHE	Partial	Full on-device execution; multi-stream design
Liberate.FHE	GPU (Python/CUDA)	CKKS	Partial	Native multi-GPU; used by recent private transformers
CPU acceleration	Vector ISA	Library-dependent	—	Vectorized modular arithmetic; degrades at high thread counts
GPU acceleration	Commodity GPU	Mainly CKKS	Design-dependent	High bandwidth; limited architectural customization

(Continued)

Table 5 (continued)

Platform	Type	Schemes	Bootstrapping	Distinguishing Characteristic
FPGA acceleration	Reconfigurable	Mainly CKKS	Design-dependent	Customizable datapaths; lower cost than ASIC
ASIC acceleration	Custom silicon	Scheme-specific	Design-dependent	Highest performance; cluster and chiplet designs for scalability

Note: Evaluations of hardware platforms have concentrated on small convolutional models, so their behavior on transformer-scale workloads is not yet well established.

3.5 Verifiable Outsourcing and Zero-Knowledge-Augmented Homomorphic Encryption

Homomorphic encryption guarantees confidentiality but says nothing about correctness: a server that returns a well-formed ciphertext of the wrong value is indistinguishable, to the client, from an honest one. In the semi-honest model this is unproblematic, but in the outsourcing settings where HE is most attractive the server is often precisely the party one is unwilling to trust. Zero-knowledge proofs supply the missing integrity guarantee, allowing a prover to convince a verifier that a computation was performed correctly without revealing the witness. Applied to machine learning, zkCNN demonstrated succinct zero-knowledge proofs for convolutional-network predictions and for claimed accuracy on a private test set, making it possible to publish a verifiable claim about a model without disclosing its weights [10], and subsequent work has extended proof systems from inference to the far more demanding case of training, establishing the practical parameters under which a party can prove that a released model is the result of honestly executing a specified training procedure on committed data [11].

The combination of proofs with encryption is natural because the two address orthogonal failure modes: encryption prevents the server from learning the input, and a proof prevents it from lying about the output. Frameworks that pair verifiable computation with homomorphic encryption make this pairing concrete, generating proofs over encrypted operations so that a client obtains both confidentiality and assurance of correctness [65]. The principal obstacle is cost. Proving a computation is typically orders of magnitude more expensive than performing it, and proving a homomorphic computation compounds two already expensive primitives, which is why current systems restrict proofs to the parts of a pipeline where integrity is most consequential rather than proving the whole. We return to this composition cost in Section 10 and to its deployment implications in Section 12.

3.6 Comparative Perspective

Relative to the other confidentiality technique surveyed here, secure multi-party computation, homomorphic encryption occupies the low-communication, high-computation corner of the design space: it requires a single round in the simplest client–server setting and therefore tolerates high-latency networks well, but it pays for this with ciphertext expansion and arithmetic that remains orders of magnitude slower than plaintext even after hardware acceleration. Against differential privacy it is complementary rather than competing, since it protects inputs and leaves the output untouched, whereas differential privacy does the reverse; the two are combined in the approximate-decryption remedy described in Section 3.1 and in the federated architectures of Section 7.3. Against trusted execution environments it trades a hardware trust assumption for a computational cost. The practical implication, developed further in Sections 10 and 11, is

that homomorphic encryption is the technique of choice when a single untrusted server must compute on data it may not see, when round-trip interaction is impossible or expensive, and when per-query latency of seconds to minutes is tolerable.

4 Differential Privacy in Deep Learning and Data Analytics

Differential privacy (DP) provides a mathematical definition of privacy that is robust to arbitrary auxiliary information and degrades gracefully under composition. A randomized mechanism is (ε, δ)-differentially private if, for any two datasets differing in a single record and any set of outcomes, the probability of producing an outcome in that set changes by at most a multiplicative factor governed by ε plus an additive term δ. The parameter ε is the privacy budget: smaller values mean stronger privacy and, typically, lower utility [18,19].

4.1 Foundations

The foundational mechanisms add noise calibrated to a query's sensitivity—the maximum change its output can exhibit when a single record changes. The Laplace and Gaussian mechanisms achieve pure and approximate DP respectively, and the exponential mechanism extends the framework to non-numeric outputs [18]. The comprehensive treatment by Dwork and Roth established the algorithmic foundations, including the central role of sensitivity, the basic and advanced composition theorems, and the group-privacy and post-processing properties that make DP composable and resistant to downstream manipulation [19]. A persistent practical concern, documented in critical reviews of the field, is that reported ε values are often large enough that the formal guarantee is weak, and that implementation details—clipping, sampling, and accounting choices—can silently undermine the intended protection [8].

The two workhorse mechanisms differ in the noise distribution they employ and the guarantee they yield. The Laplace mechanism adds noise scaled to the L1 sensitivity of a query and achieves pure ε-differential privacy; the Gaussian mechanism adds noise scaled to the L2 sensitivity and achieves the relaxed (ε, δ) form, which is the form used in deep learning because the L2 geometry matches that of gradient clipping [18,66]. The exponential mechanism handles selection problems where the output is a discrete choice rather than a number, sampling from a distribution that exponentially favors high-utility outcomes while preserving privacy. A property that makes DP uniquely composable among privacy notions is post-processing invariance: any computation applied to a differentially private output, without further access to the raw data, cannot weaken the guarantee, which is what allows a sanitized model to be published and used freely [19].

An influential alternative to adding noise to gradients directly is the teacher-ensemble paradigm, in which an ensemble of models trained on disjoint partitions of the private data votes, with noise added to the vote, to label public data that then trains a released student model. This shifts the privacy cost from every training step to the labeling queries and can yield favorable privacy–utility trade-offs in settings where unlabeled public data is available [67,68]. The breadth of such approaches—objective perturbation, output perturbation, gradient perturbation, and ensemble methods—reflects the fact that there is no single best way to inject privacy, and that the right choice depends on the model, the data, and the threat model.

4.2 Differentially Private Deep Learning

The dominant mechanism for private model training is differentially private stochastic gradient descent (DP-SGD), which clips each per-example gradient to bound its sensitivity, adds Gaussian noise to the aggregated gradient, and tracks the cumulative privacy loss across iterations using a tight accountant [66]. DP-SGD is now the *de facto* standard, but achieving good utility at meaningful ε requires care: practical

guidance emphasizes large batch sizes, careful learning-rate schedules, pretraining on public data, and the use of architectures and features that tolerate noise [67]. Surveys of the area document a rapid expansion of techniques—adaptive clipping, parameter-efficient fine-tuning under DP, and private generative modeling—alongside a growing recognition that the privacy–utility frontier depends as much on engineering as on the underlying mechanism [2,68]. Fig. 3 depicts the qualitative privacy–utility trade-off that governs these design choices.

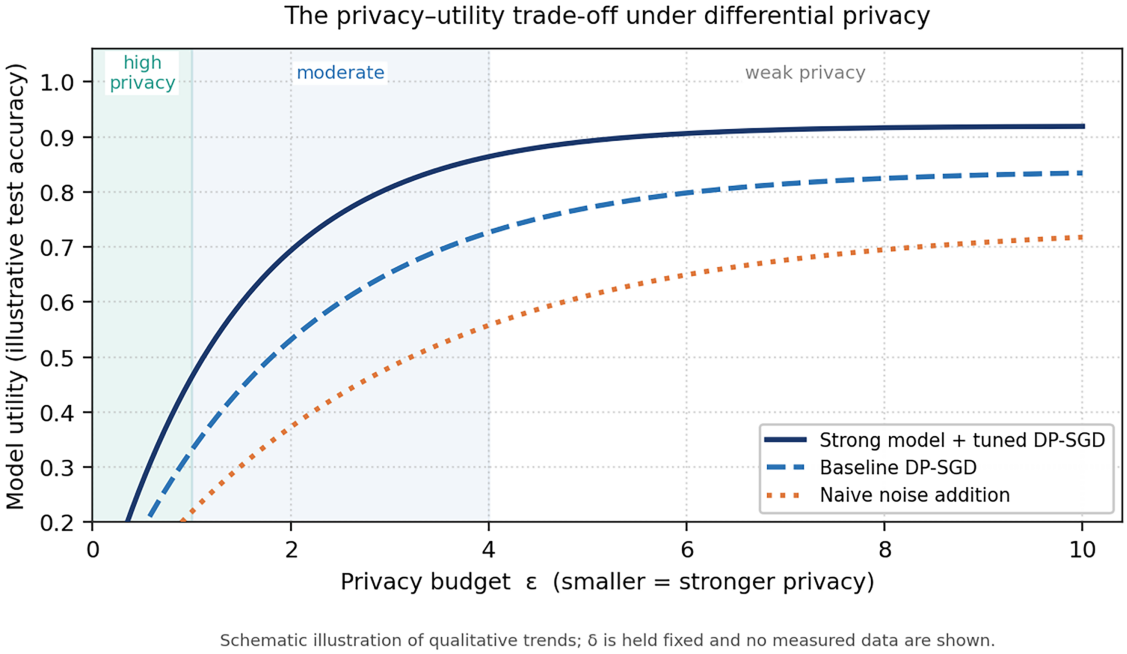


Figure 3: Schematic of the privacy–utility trade-off under differential privacy. The horizontal axis is the privacy budget ε (smaller values denote stronger privacy); the vertical axis is model utility, here illustrative test accuracy. The three curves contrast a strong architecture with tuned DP-SGD, a baseline DP-SGD configuration, and naive noise addition, showing that pretraining, architecture and accounting quality shift the entire frontier rather than merely moving along it. Shaded bands mark conventional qualitative regimes ($\varepsilon \leq 1$: strong; $1 < \varepsilon \leq 4$: moderate; $\varepsilon > 4$: weak). The curves are illustrative and do not depict measured data; δ is held fixed and is conventionally set below the inverse of the dataset size.

The mechanics of DP-SGD reward close attention because each hyperparameter has a privacy interpretation. The clipping norm bounds the contribution of any single example and therefore the sensitivity of the averaged gradient; the noise multiplier sets the ratio of Gaussian noise to that sensitivity and thus the per-step privacy cost; and the sampling rate, determined by the batch size relative to the dataset, governs privacy amplification by subsampling, whereby the random selection of each batch reduces the effective privacy loss because any given record participates only occasionally [66,67]. The interaction of these parameters with the accountant of Section 5 means that tuning DP-SGD is as much an exercise in privacy bookkeeping as in optimization.

Recent practice has shifted toward private fine-tuning of large pretrained models rather than private training from scratch, because pretraining on public data establishes useful features that DP fine-tuning can adapt with comparatively little noise, dramatically improving the achievable accuracy at a fixed budget [67,68]. Parameter-efficient methods that update only a small adapter or a low-rank correction further reduce the dimensionality over which noise must be added, compounding the benefit. These developments have narrowed, though not closed, the utility gap between private and non-private models on standard

benchmarks, and they illustrate the broader lesson that the privacy–utility frontier is moved more by what is trained, and how, than by the noise mechanism in isolation [8].

4.3 Data Analytics and Local Differential Privacy

Beyond model training, DP underpins privacy-preserving data analytics. In the central model a trusted curator holds raw data and releases noisy aggregates; in the local model, exemplified by RAPPOR, each user perturbs their own data before transmission, so that no trusted curator is required at the cost of substantially more noise [69]. The local model has seen wide industrial deployment for telemetry collection precisely because it removes the need to trust a central aggregator, aligning it naturally with the federated setting of Section 7. A central theme across both models is that the analyst must budget a finite quantity of privacy across all queries, which brings us to the problem of composition.

Differentially private synthetic data has emerged as an attractive deployment pattern for analytics because it decouples the privacy cost from the number of downstream analyses: once a synthetic dataset is generated under a fixed budget, analysts may query it arbitrarily without further privacy loss, by the post-processing property. The challenge is that generating synthetic data that preserves the multivariate structure needed for machine learning, rather than only low-order marginals, is difficult under tight budgets, and naive approaches can produce data that is simultaneously private and useless [19,68]. A complementary deployment concern is auditing: because a stated privacy budget is only as trustworthy as the implementation that produced it, empirical privacy auditing—mounting membership-inference or poisoning attacks to lower-bound the realized ε—has become an essential validation step, and discrepancies between the analytically claimed and empirically measured privacy have repeatedly exposed implementation bugs [4,8].

4.4 Differentially Private Training and Fine-Tuning of Large Language Models

The center of gravity of differentially private deep learning has shifted decisively toward large language models, and the coverage of this review would be incomplete without treating that shift explicitly. Language models are an acute privacy problem because they memorize: sequences that appear in training data can be regenerated verbatim at inference time, so a model trained on emails, clinical notes or source code is a potential disclosure channel for the individuals represented in it. Differential privacy is the principal formal defense, but applying DP-SGD naively to models with billions of parameters is unattractive because the noise added scales with the dimensionality of the updated parameter vector and because per-example gradient clipping is expensive at that scale.

Two ideas have made private language modeling practical. The first is to fine-tune rather than pretrain privately: a model pretrained on public corpora already encodes general linguistic competence, so only the adaptation to sensitive data needs protecting, and the resulting private task consumes far less budget [70]. The second is parameter-efficient adaptation, which reduces the dimension over which noise must be injected. Low-rank adaptation restricts updates to a pair of low-rank factors, so that clipping and noise addition are applied to a few million rather than a few billion parameters; combining it with DP yields better utility at a fixed ε than full private fine-tuning [71,72]. Extending this to the federated setting, on-device private fine-tuning with dynamically chosen rank has been reported to hold accuracy degradation below two percent and word-error-rate increase below seven percent at a stringent budget of $\varepsilon = 2$ with one million simulated clients [72], and framework-level designs reduce the communication burden of federated LLM adaptation by transmitting only compact adapters rather than full models [73].

A complementary and more recent line asks how private language-model quality scales with compute, data and privacy budget jointly. Empirical scaling laws for differentially private language models characterize

the frontier along these three axes simultaneously, providing guidance on how to allocate a fixed budget between more data, larger models and tighter noise, and indicating that the privacy tax diminishes as the available public pretraining corpus and compute grow [74]. Alignment procedures such as reinforcement learning from human feedback introduce a further and largely open privacy surface, because preference data is itself personal and the reward model trained on it can leak annotator judgments; extending the accounting machinery of Section 5 to multi-stage pipelines that combine private pretraining, private fine-tuning and private alignment is an active problem that the composition results surveyed there address only partially [67,68].

4.5 Comparative Perspective

Differential privacy differs from every cryptographic technique in this review in the resource it consumes: it costs accuracy rather than computation or communication, which makes it the only technique whose overhead does not grow with model size or network latency and, correspondingly, the only one whose effect is not visible in a systems benchmark. It is also the only technique that constrains what the released artifact reveals, and therefore the only defense against the membership-inference and memorization attacks of Section 1. Conversely it offers nothing against an adversary who observes the computation itself, which is why the federated systems of Section 7 pair it with secure aggregation rather than choosing between them. The practical selection rule, developed in Section 11, is that differential privacy is necessary whenever a model or statistic derived from sensitive data is published or exposed through an API, and insufficient whenever the raw data must additionally be hidden from the party performing the computation.

5 Privacy Budget Management and Composition Theorems

A single private query consumes part of a fixed privacy budget; a sequence of queries consumes more. Composition theory quantifies this accumulation, and tight composition is what makes iterative algorithms such as DP-SGD—which may execute tens of thousands of noisy gradient steps—practically private at acceptable utility. Loose accounting wastes budget and forces excessive noise; tight accounting is therefore of direct practical consequence [19].

5.1 Composition Theorems

Basic composition simply sums the ε and δ parameters of the constituent mechanisms, but this bound is far from tight for many independent or adaptively chosen mechanisms. The advanced composition theorem shows that the privacy loss of k-fold composition grows roughly with the square root of k rather than linearly, a consequence of boosting-style concentration arguments [75]. The optimal composition theorem characterizes the exact worst-case behavior of composing (ε, δ)-mechanisms, although obtaining the exact result is difficult in general [76]. These results collectively explain why naively adding budgets dramatically overstates the true privacy cost of a long training run.

The intuition behind the square-root improvement of advanced composition is that privacy losses across independent mechanisms behave like a random walk rather than a deterministic sum: although the worst-case loss of each step adds, the typical loss concentrates, and over k steps the high-probability loss grows with the square root of k. Formalizing this requires accepting a small failure probability, which is the origin of the δ term in (ε, δ) accounting [75]. The optimal composition theorem sharpens this to the exact constant but reveals that exact computation is hard in the general heterogeneous case, motivating the numerical methods that have since become standard [76].

5.2 Numerical Privacy Accounting

The most consequential practical advances have been numerical accountants that compute the privacy loss of the subsampled Gaussian mechanism—the core primitive of DP-SGD—to essentially arbitrary accuracy. Representing the privacy loss as a distribution and composing such distributions by convolution yields tight guarantees that earlier moment-based accountants could only bound from above [77]. Subsequent work improved the efficiency and accuracy of this approach through tighter discretization of the privacy-loss distribution [78] and through faster, adaptively evolving discretization schemes [79]. A complementary line replaces deterministic bounds with randomized estimators that convert an estimated privacy parameter into a formal guarantee with controlled error [80]. The cumulative effect is that, for the mechanisms that matter most in practice, the privacy loss of a complete training run can now be computed precisely rather than conservatively bounded, which directly translates into higher utility at a fixed privacy level.

Privacy amplification phenomena interact with composition in ways that the accountant must capture to avoid wasting budget. Amplification by subsampling, already mentioned, reduces the loss of each DP-SGD step; amplification by iteration shows that for certain contractive update rules the privacy of intermediate states improves as training continues; and amplification by shuffling establishes that randomly permuting the order in which locally private reports are processed yields a central-model guarantee far stronger than the local guarantee of any single report [81–83]. The shuffling result is especially significant because it bridges the local and central models of Section 4.3, offering much of the trust advantage of local privacy with utility closer to the central model, and it underpins several deployed analytics systems.

The maturation of numerical accounting has had a second-order effect that is easy to overlook: it has made privacy budgets comparable across systems only to the extent that everyone uses the same accountant, and divergent accounting choices remain a source of apparent but illusory differences in reported privacy. Two systems that add identical noise can report different ε values simply because one uses a Rényi-based bound and the other a privacy-loss-distribution bound, and a reader who does not check the accounting method may wrongly conclude that one mechanism is more private than the other [77,78,84]. This is why the field has moved toward shared, audited accounting libraries as community infrastructure, and why honest reporting practice now includes stating not only the budget but the accountant and its parameters. The convergence on common tooling is as important to cumulative progress as the underlying mathematical advances [67,79].

5.3 Relaxations: Rényi and Gaussian Differential Privacy

Several relaxations of the original definition were introduced specifically to make composition clean. Rényi differential privacy measures privacy loss using the Rényi divergence, which composes by simple addition across orders and yields tight bounds for the Gaussian mechanism [84]. Gaussian differential privacy, a special case of the broader f-differential privacy framework, characterizes privacy through the hypothesis-testing trade-off between two shifted Gaussians; it preserves an interpretable testing semantics, composes losslessly in an algebraic fashion, and admits a central-limit phenomenon under which the composition of many mechanisms converges to a Gaussian privacy profile [85]. These relaxations are not merely theoretical conveniences: they are the language in which modern accountants are expressed, and the choice among them materially affects the noise that a system must add to hit a target budget.

The choice among accounting frameworks is not purely aesthetic, because each yields a different bound for the same mechanism and therefore prescribes a different amount of noise to reach a target budget. Rényi accounting is convenient and composes trivially but can be loose when converted back to the (ε, δ) form that practitioners report; the privacy-loss-distribution and Gaussian-differential-privacy frameworks are tighter for the subsampled Gaussian mechanism but more involved to compute [77,84,85]. In practice, mature

libraries implement several accountants and report the tightest applicable bound, and the steady tightening of these bounds over the past five years has translated directly into accuracy gains for private models at no cost in privacy—an unusually clean example of theory improving practice [78,79].

5.4 Empirical Privacy Auditing

Composition theory and numerical accountants produce an upper bound on privacy loss derived from the algorithm as specified. Nothing in that analysis guarantees that the implementation matches the specification, and the history of the field contains numerous instances in which a subtly incorrect clipping, sampling or noise-generation step voided a claimed guarantee. Empirical privacy auditing closes this gap by producing a lower bound: an adversary is instantiated, canary records are planted, and the adversary's distinguishing success is converted into a statistically valid statement that the mechanism has a privacy loss of at least some value ε. When the empirical lower bound approaches the analytical upper bound, the analysis is tight and the implementation is probably correct; when they diverge, either the analysis is loose or the implementation is flawed, and both conclusions are actionable [86].

The practical obstacle to auditing was long its cost, since obtaining a tight lower bound classically required retraining the model hundreds or thousands of times. The decisive advance was the observation that many canaries can be inserted independently within a single training run and audited jointly, exploiting the connection between differential privacy and statistical generalization to avoid the loss inherent in group privacy; this reduces the cost of an audit to one training run and makes routine auditing of production pipelines feasible [87]. Subsequent work has tightened the resulting estimates by auditing against the hypothesis-testing characterization of privacy rather than the (ε, δ) parameterization, exploiting the same single-run structure while producing bounds that are closer to the true privacy profile [85,88]. Auditing has also proved diagnostically valuable beyond validation: the persistent gap between empirical lower bounds and analytical upper bounds in the black-box setting, where the adversary sees only the final model rather than every intermediate update, indicates that current analyses of that realistic threat model remain substantially loose, and closing it is among the more consequential open problems in the area [80,86].

5.5 Comparative Perspective

Privacy accounting has no analogue among the cryptographic techniques surveyed here, and the asymmetry is instructive. A cryptographic protocol either satisfies its security definition or does not; there is no cumulative budget that erodes with use, and repeated invocation of a secure protocol does not weaken it. Differential privacy is unique in degrading under composition, which is simultaneously its principal practical burden and the source of its most useful property, namely that the degradation is quantifiable in advance. The auditing machinery described above is likewise unique: no comparable empirical procedure exists for placing a lower bound on the information leaked by a homomorphic-encryption deployment or a searchable-encryption scheme, and the absence of such a procedure is precisely what makes the leakage quantification problem of Section 8.2 so difficult. Importing the auditing methodology into the leakage setting is, in our assessment, one of the more promising cross-pollinations available to the field.

Auditing and proving are complementary responses to the same question of whether a deployed mechanism does what it claims. An audit produces statistical evidence after the fact and requires trust in whoever ran it; a zero-knowledge proof of correct execution produces a verifiable guarantee in advance but at substantially higher cost and only for properties that can be expressed as a circuit. For differential privacy the natural division is that proofs can attest to the mechanical facts—that gradients were clipped to the stated norm and that noise of the stated scale was drawn from a committed source—while auditing

remains necessary for the end-to-end claim, since no proof system establishes that the privacy analysis itself is correct [11,87].

6 Secure Multi-Party Computation Protocols

Secure multi-party computation (MPC) allows a set of parties, each holding private inputs, to jointly evaluate a function so that every party learns the output and nothing else. Where homomorphic encryption concentrates the computation on a single server holding ciphertexts, MPC distributes both data and computation across parties, trading the heavy public-key arithmetic of HE for rounds of communication and large volumes of correlated randomness [1].

6.1 Foundations

The field originates with Yao's garbled circuits, which allow two parties to evaluate any Boolean circuit securely by having one party encrypt the circuit's truth tables and the other evaluate them obliviously [89]. Secret-sharing approaches in the arithmetic setting, of which the SPDZ protocol is the canonical dishonest-majority example, precompute input-independent correlated randomness in an offline phase using somewhat-homomorphic encryption and then consume it in a fast online phase, yielding active security against a dishonest majority [17]. These two paradigms—garbled circuits for Boolean and bit-level operations and secret sharing for arithmetic—are combined in modern mixed-protocol frameworks that switch representations to suit each sub-computation, and a recent survey catalogs the concretely efficient protocols that result [9].

Beneath the protocol-level abstractions lie a small number of cryptographic primitives whose efficiency determines everything above them. Oblivious transfer, in which a receiver obtains exactly one of a sender's messages without revealing which and without learning the others, is complete for secure computation, and oblivious-transfer extension makes it cheap by bootstrapping a small number of public-key operations into arbitrarily many symmetric-key ones. Beaver multiplication triples—precomputed shared products consumed to evaluate multiplications in the online phase—are the arithmetic analogue, and the offline generation of triples is where dishonest-majority protocols such as SPDZ concentrate their public-key cost [9,17]. Function secret sharing has more recently provided a route to very low online communication for comparison and selection, which are the operations that dominate machine-learning workloads such as ReLU evaluation.

6.2 Multi-Party Computation for Machine Learning

Specializing MPC to machine learning, SecureML established efficient two-party protocols for training linear models, logistic regression, and neural networks using a mixture of arithmetic sharing and garbled circuits [16]. ABY3 extended this to three-party honest-majority computation with efficient conversions among arithmetic, Boolean, and Yao sharings, a design that has become a template for subsequent systems [15]. Performance has since improved through GPU acceleration: Piranha provides a platform that ports the local operations of secret-sharing protocols onto graphics processors, reducing the time for private convolutional-network inference from hours to minutes [90], and high-throughput frameworks such as PIGEON push non-linear operations and large-batch ImageNet inference substantially further [91]. Usability has improved in parallel: SecretFlow-SPU compiles standard machine-learning programs to MPC backends, allowing practitioners to write ordinary code that executes under cryptographic protection [92].

The comparison operation deserves special mention because it is both ubiquitous in machine learning—every ReLU, max-pool, and argmax reduces to comparisons—and disproportionately expensive in MPC, since comparison is a bit-level operation poorly suited to arithmetic sharing. A large fraction of the

engineering in systems such as SecureML and ABY3 is devoted to evaluating comparisons efficiently, whether by switching to a Boolean representation, by using specialized comparison protocols, or by approximating the non-linearity with a cheaper function [15,16]. The GPU-accelerated systems address the complementary bottleneck of the linear-algebra-heavy local computation, which maps naturally onto data-parallel hardware once the protocol's communication structure is fixed [90–92].

The progression of two-party secure inference protocols illustrates how steadily the field has reduced the cost of the operations that machine learning stresses most. Early systems combined homomorphic encryption for linear layers with garbled circuits for activations, paying a large communication cost for each non-linearity; subsequent protocols introduced specialized, communication-efficient subprotocols for the comparison and truncation operations that dominate fixed-point neural-network evaluation, and integrated them into mixed-protocol frameworks that switch representations automatically [9,15]. The accumulated effect of these optimizations, compounded by the GPU acceleration discussed above, is that private inference for convolutional networks has moved from a research curiosity measured in hours to a service measured in seconds, even as the privacy guarantee—that neither party learns the other's secret—has remained unchanged [90,91]. The same trajectory is now being retraced for transformers, one non-linearity at a time.

6.3 Secure Transformer and Language-Model Inference

As with homomorphic encryption, the transformer architecture is the current proving ground for MPC. MPCFormer demonstrated fast and private transformer inference by replacing expensive non-linear operations with MPC-friendly approximations and recovering accuracy through knowledge distillation [93]. The hybrid HE/MPC systems discussed in Section 3.3—BOLT, BumbleBee, and the three-party PUMA—can equally be read as MPC contributions, since their efficiency hinges on the design of secure protocols for softmax, GELU, and matrix multiplication [23,51,54]. The convergence of the HE and MPC literatures around transformer inference illustrates a broader trend: the boundary between the two paradigms has blurred, and the most effective systems combine homomorphic packing of linear layers with interactive evaluation of non-linearities.

A structural observation unifies the secure-inference literature across the HE and MPC camps: every system is, at bottom, an exercise in allocating each layer of a network to the cryptographic primitive that evaluates it most cheaply, and then minimizing the cost of converting representations between layers. Linear layers—matrix multiplications and convolutions—are cheapest under homomorphic packing or arithmetic secret sharing; comparisons and piecewise-linear activations are cheapest under garbled circuits or function secret sharing; and smooth non-linearities are handled either by polynomial approximation or by interactive subprotocols [23,54,93]. The conversions between these regimes—share-to-ciphertext, arithmetic-to-Boolean—carry their own cost, and a recurring research theme is reducing the number and price of these conversions. Seen this way, the apparent diversity of systems resolves into different answers to a single allocation problem, which is why advances in one camp so readily transfer to the other [52,53].

Secure training, as opposed to inference, remains substantially harder and less mature. Training requires backward passes whose multiplicative depth and numerical range dwarf those of inference, and the privacy requirements are more demanding because the training data, not merely a single query, is at stake. Most deployed systems therefore target inference, with training addressed through the federated and differentially private approaches of the next section rather than through pure cryptographic computation [16,58]. Bridging this gap—practical, fully cryptographic training of large models—is among the field's most ambitious open goals.

6.4 Comparative Perspective

Secure multi-party computation is the natural counterpart to homomorphic encryption, and the choice between them is usually decided by the network rather than by the cryptography. MPC replaces expensive public-key arithmetic with cheap symmetric operations plus communication, so it is markedly faster than HE when the parties enjoy high-bandwidth, low-latency links, as in a data center, and markedly slower when they do not, as across the public internet or to mobile devices. It also assumes something HE does not, namely that a stated number of parties will not collude, which is an organizational rather than a mathematical guarantee and must be justified by the deployment context. Against federated learning, MPC is not an alternative but a component: secure aggregation, discussed in Section 7.1, is an MPC protocol specialized to summation. The recurring lesson of Section 6.3, and the reason the HE and MPC literatures have converged on transformer inference, is that the most efficient systems are hybrids that assign each layer to whichever primitive evaluates it most cheaply and then minimize the cost of converting between representations.

Zero-knowledge techniques also underpin the standard route from semi-honest to malicious security in multi-party computation. Rather than redesigning a protocol, one compiles it by requiring each party to prove, without revealing its shares, that every message it sends is consistent with its committed inputs and the protocol specification; the authenticated-sharing and message-authentication-code machinery of dishonest-majority protocols is a lightweight specialization of this idea, and full proof systems are reserved for the steps where cheating would be most damaging [11,17,94].

7 Privacy-Preserving Federated Learning

Federated learning (FL) trains a shared model across many clients that retain their raw data locally, exchanging only model updates with a coordinating server. Federated averaging, in which clients perform local gradient steps and the server averages the resulting models, is the workhorse algorithm and the starting point for the field [14,95]. FL reduces data movement, but as the gradient-inversion results below show, the updates it exchanges can themselves leak the underlying data, so FL is best understood not as a privacy solution in itself but as a framework into which cryptographic and perturbative protections must be integrated.

7.1 Secure Aggregation

Secure aggregation allows the server to learn the sum of client updates without learning any individual update. The foundational protocol masks each client's update with pairwise-cancelling random values and uses secret sharing to recover from client dropouts, providing security under both semi-honest and malicious-server models [96]. Because its communication scales poorly with the number of clients, subsequent work reduced the per-client overhead from linear to polylogarithmic by replacing the complete communication graph with a sparse random graph [97], and lightweight designs further reduced cost through one-shot reconstruction of the aggregate mask [98]. For the cross-silo setting with a small number of well-resourced participants, homomorphic encryption offers an alternative: BatchCrypt packs and quantizes gradients before encrypting them, reporting training speedups of twenty-three to ninety-three times and communication reductions of sixty-six to one hundred times relative to naive per-value encryption, at under one percent accuracy loss [99]. Multi-key homomorphic encryption extends this to settings where no single party may hold the decryption key [100], and integrated systems such as FedML-HE provide end-to-end homomorphic aggregation [101].

Federated learning is usually divided into two regimes with markedly different constraints. In cross-device federated learning, the participants comprise vast numbers of intermittently available, resource-constrained devices such as phones, so protocols must tolerate frequent dropouts, scale to millions of clients,

and minimize on-device computation and communication; this is the regime for which the polylogarithmic and lightweight secure-aggregation protocols were designed [97,98]. In cross-silo federated learning, a handful of well-resourced organizations—hospitals, banks—collaborate, so the number of parties is small and stable but each holds a large, valuable dataset, which makes the heavier homomorphic-encryption approach with its strong confidentiality attractive [99,101]. Matching the privacy mechanism to the regime is a primary design decision.

7.2 Gradient-Leakage Attacks and Defenses

The motivation for these cryptographic protections is the surprising strength of gradient-inversion attacks. Deep leakage from gradients showed that, by optimizing synthetic inputs to match the shared gradient, an adversary can reconstruct training images and labels with pixel-level fidelity [5]. The inverting-gradients attack improved fidelity using a magnitude-invariant cosine objective and demonstrated reconstruction from trained deep networks and from gradients averaged over several examples [6]; broader surveys of privacy attacks catalog how gradient-leakage, membership-inference, and model-inversion attacks together form a coherent threat landscape against machine-learning models [102]. These results refute the intuition that sharing gradients rather than data confers privacy. Defenses fall into three families that map onto the techniques of this review: adding differentially private noise to updates, cryptographically hiding individual updates via secure aggregation or HE, and gradient compression or sparsification, which both reduces communication and degrades the information available to an attacker [103].

The evolution of gradient-leakage attacks carries a methodological lesson that extends beyond federated learning: privacy claims based on the intuition that a representation "does not look like" the original data are unreliable, because optimization can often invert the representation. The earliest attacks succeeded only on single examples and small networks; successive refinements incorporated image priors, exploited the auxiliary statistics that batch normalization leaks, and extended to batches and to trained rather than randomly initialized models, steadily widening the regime in which reconstruction succeeds [5,6,102]. The practical implication is that the defender cannot rely on the apparent abstraction of gradients but must add a mechanism with a provable guarantee—cryptographic hiding, differentially private noise, or aggressive compression—and must assume that any information left in the clear will eventually be extracted. This is the same lesson that the leakage-abuse attacks of Section 8 teach for searchable encryption, and it is one of the strongest empirical regularities in the field [103,104].

7.3 Combining Federated Learning with Differential Privacy and Encryption

Robust federated systems layer protections. User-level differential privacy bounds what the final model reveals about any client's entire local dataset, and careful analysis of clipping and noise allocation optimizes the resulting privacy–utility trade-off [105]. Distributed noise generation reconciles DP with secure aggregation: the distributed discrete Gaussian mechanism allows each client to add a share of the required noise so that the aggregate is correctly calibrated even though no party adds it centrally [104]. Communication-efficient private FL frameworks unify compression with privacy accounting to reduce both bandwidth and budget consumption [103], and homomorphic encryption can be combined with FL to protect updates from the server entirely, as cataloged in dedicated surveys [22,106]. Systematic reviews of differentially private FL document the design space and the recurring tension between client-level and example-level guarantees [21].

The layered architectures that combine federated learning with differential privacy and encryption also force a careful accounting of where trust resides at each stage. Secure aggregation removes trust in the server's handling of individual updates but not trust concerning what the final model discloses; differential privacy at the client removes the need to trust the aggregator with raw updates but injects more noise; distributed

noise generation splits the trust by having clients jointly produce a correctly calibrated total amount of noise without any single party controlling it; and homomorphic aggregation shifts trust onto the cryptographic assumption and the key-management arrangement [96,100,104]. A complete system specifies, for each asset and each stage, which party is trusted with what, and a privacy claim is only as strong as the weakest such assumption. Making these trust assumptions explicit—rather than implicit in the choice of mechanism—is, in our view, an underappreciated discipline that the most careful deployments already practice and that the literature would benefit from standardizing [21,106].

7.4 Byzantine Robustness and Personalization

Privacy is not the only adversarial concern in federated learning; robustness to malicious clients who submit poisoned updates is equally important, and the two goals can conflict, since cryptographic hiding of updates also hides evidence of misbehavior. Robust aggregation rules such as Krum, which selects the update closest to its neighbors, and coordinate-wise median and trimmed-mean estimators tolerate a bounded fraction of Byzantine participants [107,108], while trust-bootstrapping approaches such as FLTrust use a small clean root dataset to score client updates [109]. Recent work explicitly reconciles differential privacy with Byzantine resilience, showing that both can be achieved simultaneously with careful protocol design [110]. A parallel line addresses statistical heterogeneity through personalization, tailoring models to each client's distribution while sharing common structure, as surveyed comprehensively in the literature [111].

The tension between privacy and robustness is fundamental rather than incidental. Robust aggregation rules inspect individual client updates to detect outliers, but the very secure-aggregation and homomorphic-encryption mechanisms that protect privacy hide those updates from inspection, so a server cannot simultaneously verify and remain blind to client contributions. Reconciling the two has spawned protocols that perform robust statistics inside the cryptographic computation—computing a median or a trust score over secret-shared or encrypted updates—at additional cost, and the explicit joint treatment of differential privacy with Byzantine resilience shows that the two goals, though in tension, are not irreconcilable [107,108,110]. A further practical axis is the federation topology: in addition to the horizontal partitioning assumed above, where parties hold different samples with the same features, vertical federation—where parties hold different features of the same samples—arises in financial and healthcare consortia and demands its own privacy-preserving protocols for entity alignment and split computation [1,101].

7.5 Verifiability and Zero-Knowledge Proofs in Federated Learning

Section 7.4 described the tension between privacy and robustness: cryptographic hiding of client updates also hides evidence of misbehavior. Zero-knowledge proofs offer a principled resolution, because they allow a client to demonstrate that its hidden update satisfies a stated predicate without revealing the update itself. The most direct application is input validation for secure aggregation, where each client proves that its masked contribution is well-formed and bounded in norm, so that the server can reject poisoning attempts that would otherwise be concealed by the very masking that protects honest clients; protocols of this kind add input validation to secure aggregation at a cost that scales acceptably with the number of clients [96,112].

A second application concerns the integrity of the aggregation itself. In a conventional federated protocol clients must trust the server to aggregate faithfully, and a malicious server can bias the global model or single out a client's contribution. Schemes based on succinct non-interactive arguments allow the server to publish a proof that the released global model is the correct aggregate of the received updates, with the proof verifiable by any participant and optionally anchored on a public ledger for auditability [113]. Combining such proofs with homomorphic aggregation yields a protocol in which the server neither sees individual updates

nor is trusted to combine them correctly [65,101]. The costs remain substantial, and current deployments therefore reserve proofs for cross-silo federations, where participants are few, well resourced and mutually suspicious, rather than for cross-device settings with millions of phones.

7.6 Federated Learning for Large Language Models

Federated learning was designed for models small enough to be trained on a phone, and the arrival of models with billions of parameters has strained every assumption of the original design. Transmitting a full model update per round is infeasible on consumer connections, on-device memory cannot hold optimizer state for a large model, and the statistical heterogeneity of user data interacts badly with the sharpness of large-model optimization. The response has been to federate adaptation rather than training: clients fine-tune a small adapter over a frozen public backbone and transmit only the adapter, reducing communication by orders of magnitude and making the privacy accounting tractable because the noise-perturbed update is small [71,72]. Framework-level designs decouple the client-side model from the full server-side model so that clients never need to hold the complete network [73].

This convergence of federated learning with parameter-efficient adaptation has an important consequence for privacy analysis. The low-rank structure that makes federated LLM adaptation feasible also concentrates the information content of an update, which cuts both ways: fewer parameters mean less noise for a given budget, but they may also mean that each parameter carries more information about the client's data, so gradient-inversion analyses developed for full updates do not transfer unchanged. Establishing the leakage profile of adapter-only updates, and determining whether the secure-aggregation and distributed-noise machinery of Sections 7.1 and 7.3 composes correctly with low-rank structure, is in our view among the most pressing open questions raised by this literature [21,106].

7.7 Comparative Perspective

Federated learning differs from the other techniques surveyed in that it is an architecture rather than a guarantee. It relocates computation without, by itself, bounding what the relocated messages reveal, as the gradient-inversion results of Section 7.2 establish decisively. Its correct place in a comparison is therefore as the setting into which the other techniques are inserted: secure aggregation or homomorphic encryption to hide individual updates, differential privacy to bound what the released model reveals, zero-knowledge proofs to restore the integrity that hiding removed, and robust aggregation to tolerate malicious participants. Judged against centralized training with differential privacy, federated learning offers a genuine advantage in data governance, since raw data never leaves its jurisdiction, at the cost of higher communication, weaker convergence under heterogeneity, and a substantially larger attack surface. That governance advantage, rather than any cryptographic superiority, is what drives its adoption in the regulated settings examined in Section 12.

8 Encrypted Search and Database Systems

Machine-learning pipelines depend on data stores, and outsourcing those stores to untrusted infrastructure raises its own privacy problem: how to search and query encrypted data without revealing its contents or the queries themselves. This area supplies the storage-layer foundations on which privacy-preserving learning systems are built [24].

8.1 Searchable Symmetric Encryption

Searchable symmetric encryption (SSE) enables a client to outsource an encrypted document collection and later issue encrypted keyword queries that the server can evaluate without learning the documents

or, ideally, the keywords. The first practical scheme established the feasibility of searching encrypted data with manageable overhead [114], and subsequent foundational work formalized the security definitions and introduced efficient index-based constructions with sublinear search [115]. Modern dynamic SSE schemes add forward and backward privacy—ensuring that newly inserted documents cannot be linked to past queries and that deleted documents become unsearchable—and recent work has both rethought the security foundations of SSE and pushed practical constructions toward richer query types such as arbitrary Boolean queries over joined tables [25,116,117]. A current survey provides a comprehensive map of the design space and its security notions [24].

The security of dynamic searchable encryption is characterized through two properties that constrain what updates may reveal. Forward privacy requires that an addition to the database cannot be linked to previously issued search tokens, which thwarts injection attacks that would otherwise exploit the server's ability to test new documents against old queries; backward privacy requires that searches do not reveal information about documents that have already been deleted [25,118]. Achieving both efficiently, while supporting expressive queries such as conjunctions and Boolean combinations over joined relations, is the central technical challenge of modern SSE, and recent constructions have made substantial progress on the trade-off between expressiveness, efficiency, and leakage [24,116,117].

8.2 Leakage-Abuse Attacks

SSE schemes deliberately leak some information—typically access patterns and query repetition—in exchange for efficiency, and a sustained line of research shows that this leakage can be devastating. Leakage-abuse attacks against forward- and backward-private schemes recover queried keywords by correlating observed access patterns with auxiliary knowledge of the document distribution [118], and volumetric injection attacks recover queries by inserting crafted documents and observing the resulting changes in response volume, with recent work characterizing the optimal injection size [119]. These attacks establish that a scheme's leakage profile, not merely its cryptographic soundness, determines its real-world security, and they motivate padding and volume-hiding countermeasures such as those implemented in encrypted document databases [120].

Leakage-abuse attacks can be organized by the leakage they exploit and the capability they assume. Access-pattern attacks observe which encrypted documents each query touches and, given partial knowledge of the document distribution, recover the underlying keywords; volume attacks observe only the number of results and exploit the distinctiveness of result-set sizes; and injection attacks assume the stronger capability of inserting chosen documents into the database, which lets the adversary engineer distinctive leakage and recover queries with high precision [118,119]. The countermeasures trade efficiency for leakage suppression along a spectrum: padding result sets to fixed sizes blunts volume attacks at the cost of bandwidth; volume-hiding constructions and oblivious data structures raise the cost further; and full obliviousness via ORAM eliminates access-pattern leakage entirely at the highest cost [120,121]. Choosing a point on this spectrum requires quantifying the adversary's realistic capabilities, which is exactly the kind of leakage accounting for which the field still lacks a unified theory.

8.3 Oblivious RAM and Encrypted Databases

The strongest defense against access-pattern leakage is oblivious RAM (ORAM), which makes any two access sequences of equal length computationally indistinguishable by continuously reshuffling and re-encrypting storage. Path ORAM achieves this with a simple tree-based construction and logarithmic bandwidth overhead, and it remains the most widely deployed scheme [121]. Recent advances add integrity to optimal ORAM constructions [122] and reduce bandwidth in the distributed and multi-party settings

relevant to secure computation [123]. At the system level, encrypted databases such as CryptDB demonstrated that practical SQL query processing over encrypted data is achievable by layering property-revealing encryption schemes, establishing a template that later systems have refined with stronger leakage control [124]. The throughline of this section is that storage-layer privacy is governed by the same leakage-vs.-efficiency trade-off that pervades the rest of PPML.

Oblivious RAM is not the only route to access-pattern protection, and the alternatives illuminate the design space. Private information retrieval allows a client to fetch a record from a database without the server learning which record, using homomorphic encryption or multi-server replication rather than the continuous reshuffling of ORAM, and is well suited to read-only lookup workloads such as private inference over a public model [7]. Trusted execution environments offer a hardware-based alternative in which computation occurs inside an attested enclave, sidestepping cryptographic overhead at the cost of trusting the hardware vendor and contending with a long history of side-channel attacks against enclaves. Structured encryption generalizes searchable encryption to arbitrary data structures—graphs, matrices—enabling encrypted queries beyond keyword search, which is increasingly relevant as machine-learning pipelines query graph-structured and relational data [24,25]. The persistent lesson across all these mechanisms is that access patterns are a powerful side channel, and that protecting the contents of data without protecting the pattern of access to it provides incomplete privacy [118,119].

8.4 Comparative Perspective

Encrypted search occupies a distinctive position because it is the only technique surveyed here whose security definition explicitly concedes a quantified leakage rather than aiming to eliminate it. This makes direct comparison with homomorphic encryption misleading: a searchable-encryption scheme is far faster than an equivalent homomorphic computation precisely because it reveals access and volume patterns that homomorphic evaluation would hide, and the sustained record of leakage-abuse attacks documented in Section 8.2 shows that this concession is not merely theoretical. The comparison that matters is therefore not speed but the relationship between an application's realistic adversary and the specific leakage a scheme permits. Oblivious RAM sits at the other end of the same spectrum, eliminating access-pattern leakage at logarithmic bandwidth cost, and private information retrieval offers a third point suited to read-only workloads. For machine-learning pipelines the practical significance is that the data layer is frequently the weakest link: a training pipeline whose gradients are protected by secure aggregation but whose feature store leaks access patterns has simply relocated the vulnerability.

The same integrity gap appears in outsourced storage. A searchable-encryption server that returns an incomplete result set is not detectable by the leakage analysis, which characterizes what the server learns rather than whether it answers honestly, and a client that trains a model on silently truncated query results inherits a corrupted dataset with no indication of the fact. Verifiable searchable encryption addresses this by attaching proofs of completeness to query responses, closing the loop between the confidentiality guarantees of Section 8.1 and the integrity guarantees discussed in Sections 3.5 and 7.5 [10,24].

9 Post-Quantum Cryptographic Solutions

All of the cryptographic techniques surveyed above rest on computational hardness assumptions, and the advent of large-scale quantum computers threatens the most widely used of these. Ensuring that privacy-preserving systems remain secure in a post-quantum world is therefore not a separate concern but a foundational requirement for every cryptographic component of PPML [125].

9.1 The Quantum Threat

Shor's algorithm solves integer factorization and discrete logarithms in polynomial time on a quantum computer, breaking the RSA and elliptic-curve cryptosystems that secure most of today's communications and key exchange [20]. The threat is not confined to the future: in the harvest-now-decrypt-later model, an adversary records encrypted traffic today and decrypts it once a sufficiently capable quantum computer exists, which means that data requiring long-term confidentiality must be protected with quantum-resistant cryptography now. Lattice-based cryptography, the same mathematical foundation that underlies the homomorphic-encryption schemes of Section 3, is widely believed to resist quantum attack, giving HE a natural post-quantum character [7,125].

It is important to distinguish which cryptographic primitives are threatened and which are not. Shor's algorithm breaks the public-key primitives based on factoring and discrete logarithms—RSA, Diffie–Hellman, and elliptic-curve cryptography—that secure key exchange and signatures [20]. Symmetric primitives such as block ciphers and hash functions are weakened only quadratically by Grover's search, which is countered simply by doubling key and output lengths; this is why the security levels of the new standards are calibrated against the cost of attacking AES of various key sizes [126]. The practical consequence is that the migration burden falls on the public-key layer—precisely the layer that authenticates model updates in federated learning, establishes the channels over which encrypted queries travel, and signs the artifacts of an outsourced database—while the symmetric cores of these systems need only modest parameter increases [125].

9.2 NIST Standardization

After an eight-year process, the U.S. National Institute of Standards and Technology published its first three post-quantum standards in August 2024. FIPS 203 specifies ML-KEM, a module-lattice key-encapsulation mechanism derived from CRYSTALS-Kyber, as the primary standard for key establishment [126,127]. FIPS 204 specifies ML-DSA, derived from CRYSTALS-Dilithium, as the primary digital-signature standard [128,129], and FIPS 205 specifies SLH-DSA, a stateless hash-based signature scheme derived from SPHINCS+ that provides a conservative fallback whose security rests only on hash-function properties rather than lattice assumptions [130,131]. The standardization process subsequently selected an additional code-based key-encapsulation mechanism for the fourth round and continues to develop a lattice-based signature standard derived from Falcon [132]. Fig. 4 traces this timeline, and Table 6 summarizes the three published standards.

The eight-year standardization process is instructive for understanding the maturity and the residual uncertainty of post-quantum cryptography. From an initial pool of dozens of submissions, successive rounds of public cryptanalysis eliminated candidates—several prominent schemes were broken during the process—before the lattice-based key-encapsulation and signature schemes, together with a hash-based signature scheme as a hedge, were selected [127,129,131]. The deliberate retention of a hash-based scheme whose security rests on entirely different assumptions reflects a sober recognition that the lattice assumptions, though well studied, are younger than the factoring and discrete-logarithm assumptions they replace, and that algorithmic surprises cannot be excluded. The ongoing fourth round, which added a code-based mechanism, continues this diversification strategy, and the in-progress lattice signature standard derived from Falcon will offer smaller signatures than ML-DSA for bandwidth-constrained applications [128,132]. For privacy-preserving machine learning, the lesson is to design for cryptographic agility—the ability to swap primitives—rather than to hard-wire any single scheme.

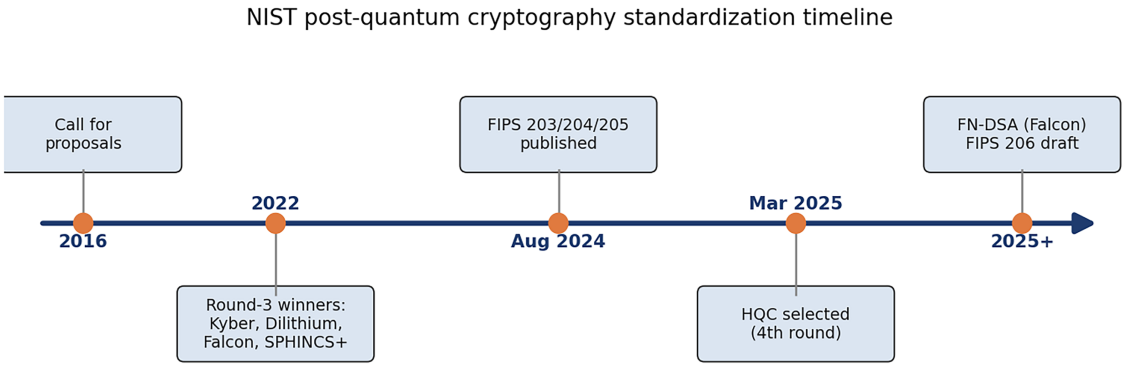


Figure 4: Timeline of the NIST post-quantum cryptography standardization process, from the 2016 call for proposals, through the 2022 announcement of the round-three selections (CRYSTALS-Kyber, CRYSTALS-Dilithium, Falcon and SPHINCS+), the August 2024 publication of FIPS 203, 204 and 205, the March 2025 selection of HQC in the fourth round, to the in-progress FN-DSA (Falcon) standard. Migration guidance and deprecation timelines associated with this process are discussed in Section 9.4.

Table 6: The three post-quantum standards published by NIST in August 2024. Each standard defines parameter sets at NIST security categories 1, 3 and 5, corresponding approximately to the classical and quantum security of AES-128, AES-192 and AES-256 respectively. Indicative sizes across those parameter sets are: ML-KEM public keys 800–1568 bytes; ML-DSA signatures 2420–4595 bytes; SLH-DSA signatures 7856–49,856 bytes. "Mathematical basis" names the hardness assumption on which security rests.

Standard	Algorithm (Origin)	Mathematical Basis	Function
FIPS 203	ML-KEM (CRYSTALS-Kyber)	Module-LWE (lattice)	Key encapsulation
FIPS 204	ML-DSA (CRYSTALS-Dilithium)	Module-LWE/Module-SIS	Digital signature
FIPS 205	SLH-DSA (SPHINCS+)	Hash-function security	Digital signature (fallback)

Note: ML-KEM is the recommended general-purpose key-establishment mechanism; ML-DSA is the recommended general-purpose signature scheme; SLH-DSA offers algorithm diversity at the cost of larger signatures.

9.3 Lattice Foundations and Integration with Privacy-Preserving Learning

The convergence of post-quantum cryptography and privacy-preserving learning is more than coincidental. The module-learning-with-errors problem that secures ML-KEM and ML-DSA belongs to the same family of assumptions as those securing CKKS, BFV, BGV, and TFHE, so a system built on lattice-based homomorphic encryption is already post-quantum at its core [30,31,127]. The practical work ahead is integration: replacing the classical key-exchange and signature components that surround a privacy-preserving computation—the transport-layer handshakes, the authentication of model updates in federated learning, the integrity of outsourced encrypted databases—with their standardized post-quantum counterparts, while managing the larger key and signature sizes that lattice and hash-based schemes entail. Table 7 contrasts the cross-cutting privacy techniques of this review along the dimensions that determine how they compose into a complete system. This table summarizes qualitative character only; Table 8 analyzes what happens when these techniques are composed.

The performance characteristics of the new standards condition how they integrate into privacy-preserving systems. ML-KEM public keys range from roughly 800 to 1568 bytes and ML-KEM ciphertexts range from 768 to 1568 bytes across security levels; ML-DSA signatures range from about 2420 to 4595 bytes; and SLH-DSA signatures are far larger, from roughly 7856 to 49,856 bytes, the price of resting security solely on hash functions [126,128,130]. These sizes, while modest for a single handshake, accumulate

in protocols that authenticate many model updates or sign many database records, and they motivate careful placement of post-quantum signatures within a federated or outsourced-storage architecture. The recommended migration pattern is hybrid: combine a classical and a post-quantum mechanism so that the channel remains secure as long as either resists attack, hedging against both quantum advances and as-yet-undiscovered weaknesses in the relatively young lattice constructions [125,132].

Table 7: Cross-cutting comparison of the principal privacy-preserving techniques surveyed. "Primary guarantee" distinguishes cryptographic confidentiality of inputs from statistical indistinguishability of outputs; "Trust model" states the assumption that must hold for the guarantee to be meaningful; "Dominant cost" names the resource the technique principally consumes.

Technique	Primary Guarantee	Trust Model	Dominant Cost
Homomorphic encryption	Cryptographic (input)	Single untrusted server	Computation/latency
Secure multi-party comp.	Cryptographic (input)	Non-colluding parties	Communication
Differential privacy	Statistical (output)	Trusted or local curator	Utility loss
Federated learning	Data locality (+ add-ons)	Honest-but-curious server	Communication/ leakage
Searchable encryption	Cryptographic (with leakage)	Untrusted storage	Leakage profile
Post-quantum primitives	Cryptographic (long-term)	Standard PKI	Key/signature size

Note: The techniques are complementary; the central design task is composing them so that one technique's residual leakage is covered by another's guarantee.

Table 8: Compatibility of the principal privacy-preserving techniques under composition. Entries characterize the dominant interaction when the row technique is combined with the column technique in a single system. "Synergistic" denotes a combination whose joint guarantee exceeds what either provides alone; "additive" denotes independent guarantees with additive cost; "tension" denotes an interaction requiring explicit joint analysis or a design compromise, discussed in the section indicated.

	Homomorphic Encryption	Differential Privacy	Secure Aggregation/MPC	Zero-Knowledge Proofs
Homomorphic encryption	—	Tension: cryptographic error and privacy noise must be analyzed jointly (Sections 3.1 and 10)	Synergistic: HE aggregation removes trust in server (Section 7.1)	Additive, costly: proofs over ciphertexts compound expense (Section 3.5)
Differential privacy	Tension (see above)	—	Synergistic: distributed noise calibrated across clients (Section 7.3)	Additive: proofs can attest correct noise generation

(Continued)

Table 8 (continued)

	Homomorphic Encryption	Differential Privacy	Secure Aggregation/MPC	Zero-Knowledge Proofs
Secure aggregation/MPC	Synergistic	Synergistic	—	Synergistic: input validation restores robustness (Section 7.5)
Federated learning	Tension: communication cost may negate federation benefit (Section 10)	Synergistic: bounds leakage of released model	Synergistic: standard pairing	Synergistic: verifiable aggregation (Section 7.5)
Robust aggregation	Tension: hiding conceals misbehavior (Section 7.4)	Tension: noise masks outliers (Section 10)	Tension: requires computation on hidden values	Synergistic: proofs replace inspection

Note: The table characterizes dominant interactions rather than exhaustive possibilities; a specific protocol may avoid a listed tension through additional design effort, invariably at some cost.

Side-channel resistance is a second integration concern that the mathematical security proofs do not address. Implementations of lattice schemes have been shown to leak secret keys through power and timing measurements during the polynomial arithmetic and sampling steps, so constant-time implementations and masking countermeasures are necessary for deployment in adversarial physical environments such as edge devices participating in federated learning [127,129]. Because much of the privacy-preserving learning stack already shares the lattice substrate of these standards, investments in hardened lattice arithmetic—constant-time number-theoretic transforms, masked samplers—pay dividends across both the homomorphic-encryption and the post-quantum-signature components of a system, reinforcing the theme that the field's foundations are more unified than its subdivisions suggest [7,125].

9.4 Migration Pathways and Crypto-Agility for Privacy-Preserving Systems

Publishing a standard is not the same as deploying it, and the migration of privacy-preserving systems to post-quantum primitives raises problems that the standards themselves do not resolve. NIST's transition guidance sets out the expected approach, identifying the quantum-vulnerable standards to be retired and the quantum-resistant standards that replace them, and proposing a timeline in which legacy public-key algorithms at the 112-bit security level are deprecated around 2030 and disallowed by 2035 [133]. For a privacy-preserving machine-learning deployment, the practical consequence is an inventory problem: every place where a classical key exchange establishes a channel, a classical signature authenticates a model update, or a classical certificate anchors trust in an aggregation server must be identified before it can be replaced, and in a federated system those places are distributed across every participating organization and device.

Two migration patterns dominate current practice. The first is hybrid deployment, in which a classical and a post-quantum mechanism are combined so that the resulting channel remains secure provided either component resists attack; this hedges against both quantum progress and undiscovered weaknesses in the comparatively young lattice constructions, at the cost of transmitting both sets of parameters. The second is crypto-agility, the architectural property that a deployed system can rotate cryptographic algorithms without redesign. Agility is the more demanding requirement and the more valuable one, because the migration

now under way will not be the last, and systems that hard-code algorithm choices, or that require firmware updates to change them, cannot respond on a relevant timescale [132,133].

For privacy-preserving machine learning specifically, the migration burden is unevenly distributed in a way that is worth stating plainly. The homomorphic-encryption and lattice-based components of the stack are already post-quantum, since they rest on the same module-learning-with-errors assumptions as ML-KEM and ML-DSA, and require no algorithmic replacement [30,127]. Secret-sharing-based multi-party computation is information-theoretically secure in its online phase and is threatened only through the public-key machinery of its preprocessing. Differential privacy is unconditionally secure and entirely unaffected. What must actually migrate is the conventional public-key infrastructure surrounding these components, together with any searchable-encryption or database deployment relying on classical public-key operations. Recognizing this asymmetry allows an organization to scope its migration realistically rather than treating the whole privacy stack as quantum-vulnerable.

9.5 Post-Quantum Considerations in Concrete Privacy-Preserving Settings

The preceding discussion is deliberately general, and the reviewer of a survey may reasonably ask what the quantum threat means for each of the architectures treated earlier in this review. The answer differs sharply by setting, and stating those differences concretely is more useful than a blanket recommendation to migrate.

In federated learning the exposure is not the model updates, which are protected by secure aggregation or homomorphic encryption whose lattice basis is already quantum-resistant, but the authentication layer. A cross-device federation authenticates millions of clients and signs a global model each round; if those signatures use elliptic-curve cryptography, an adversary with a quantum computer could forge client contributions or substitute a poisoned global model retrospectively. Migration therefore means replacing per-round signatures with ML-DSA, whose signatures are roughly thirty to seventy times larger than an equivalent elliptic-curve signature, a cost that is negligible per round for cross-silo federations but material for bandwidth-constrained devices participating in many rounds [97,128]. Where signature size dominates, the forthcoming Falcon-based standard offers a smaller alternative [132].

In outsourced encrypted search the exposure is different again and, in an important sense, more urgent. A searchable-encryption deployment stores ciphertexts that may persist for years, so the harvest-now-decrypt-later threat applies directly to the data at rest rather than merely to the channel. The symmetric core of such schemes is affected only by Grover's quadratic speed-up and is addressed by doubling key lengths, but any public-key component used for key distribution, multi-writer access control or delegation must be replaced [24,126]. A deployment that encrypts documents under a symmetric scheme but distributes keys using classical Diffie-Hellman is quantum-vulnerable in exactly the place its designers are least likely to look.

In secure multi-party computation the asymmetry is the most favorable of all. The online phase of a secret-sharing protocol is information-theoretically secure and is entirely unaffected by quantum adversaries; the vulnerability is confined to the preprocessing phase, where somewhat-homomorphic encryption or oblivious-transfer extension generates correlated randomness. Because the dominant instantiations of both are lattice- or symmetric-key-based, the practical migration burden is small, and it is chiefly the authenticated channels between parties that require attention [17,94]. For homomorphic encryption the burden is smaller still: the security parameters already assume a quantum adversary, and the relevant question is not whether to migrate but whether the chosen ring dimension and modulus leave adequate margin against improvements in lattice reduction, which is a parameter-selection problem rather than an algorithm-replacement problem [7,32].

Two cross-cutting observations follow. First, the setting with the longest confidentiality horizon is the one that most needs post-quantum treatment, and that is rarely the model-training pipeline: it involves archived data and long-lived keys. Second, differential privacy is the only mechanism surveyed whose guarantee is wholly unaffected, because it is information-theoretic; in a system whose confidentiality assumptions may expire, a differentially private release is the component that will still be sound in thirty years [18,19].

9.6 Comparative Perspective

Post-quantum cryptography is not a privacy technique in the sense that the others surveyed here are; it is a durability property that the cryptographic techniques must possess if their guarantees are to outlast the data they protect. Its comparative significance is therefore temporal. A differential-privacy guarantee, being information-theoretic, is permanent. A secret-sharing guarantee in the honest-majority online phase is likewise permanent. A homomorphic-encryption or key-exchange guarantee is computational and has an expiry date determined by cryptanalytic progress, which the harvest-now-decrypt-later threat model converts from a future concern into a present one for any data with a long confidentiality horizon. When selecting techniques for a system whose data must remain confidential for decades, as in genomics or national records, this temporal dimension should carry as much weight as latency or accuracy.

10 Compositional Compatibility and Conflicts

This review has repeatedly asserted that privacy-preserving techniques are complementary and that production systems combine them. That claim, taken alone, is too comfortable. Composition is not free, and several of the pairings most often proposed interact in ways that degrade one or both guarantees, cancel an efficiency advantage, or introduce failure modes absent from either technique alone. This section examines those interactions directly. Table 8 summarizes the pairwise picture; the discussion below treats the cases where the interaction is genuinely problematic rather than merely additive.

The most technically delicate conflict is between differential privacy and homomorphic encryption. Differential privacy requires adding noise drawn from a precisely specified continuous distribution; homomorphic encryption operates on a discrete plaintext space with a finite noise budget, and in the approximate CKKS setting already carries an error term of its own. Adding differential-privacy noise inside the encrypted domain therefore consumes multiplicative depth and interacts with the scheme's own error growth, so that the effective noise seen after decryption is the sum of a deliberate privacy term and an uncontrolled cryptographic term. If the error is correlated with the data, ignoring the cryptographic error can make the privacy analysis nonconservative; in every case, ignoring it makes the analysis inaccurate; if it is accounted for conservatively, utility suffers twice. The distributed discrete Gaussian mechanism addresses precisely this problem in the secure-aggregation setting by replacing the continuous Gaussian with a discrete analogue whose sum over clients is correctly calibrated in a finite group [104], and the approximate-decryption remedy of Section 3.1 is the same issue viewed from the cryptographic side [32]. The general lesson is that noise added for privacy and noise inherent to the cryptography must be analyzed jointly, not independently.

A second conflict is economic rather than mathematical. Federated learning is adopted principally to reduce data movement, yet the cryptographic protocols most often layered onto it increase communication, sometimes by more than the federation saves. Classical secure aggregation requires pairwise key agreement whose cost grows with the number of clients, and naive per-value homomorphic encryption of gradients inflates each update by one to two orders of magnitude. Whether the composition remains worthwhile is therefore an empirical question about the specific deployment, and the answer has driven a decade of protocol design: polylogarithmic-overhead aggregation, one-shot mask reconstruction, and batched

quantized encryption exist precisely because the naive composition negates the advantage that motivated the architecture [97–99]. A designer who adds secure aggregation to a federated system without measuring the resulting communication has not necessarily improved the system.

A third conflict, discussed in Section 7.4 and generalized here, is between confidentiality and accountability. Every mechanism that hides an individual contribution, whether secure aggregation, homomorphic encryption or local differential privacy, simultaneously removes the evidence needed to detect a malicious contribution. Robust aggregation rules require visibility into the very quantities that privacy mechanisms conceal. The resolutions available are to compute robust statistics inside the cryptographic computation at additional cost, to accept a weaker robustness guarantee, or to introduce zero-knowledge proofs so that clients demonstrate well-formedness without disclosure [11,110,112]. Each resolution has a price, and no current approach delivers strong privacy, strong robustness and low cost simultaneously.

A fourth interaction concerns the sequencing of differential privacy with cryptographic protocols. Because differential privacy is preserved under post-processing, applying it before a cryptographic aggregation is safe but wasteful, since each client must add enough noise to protect itself alone; applying it after aggregation is efficient but requires trusting whoever holds the aggregate. Distributed noise generation resolves the dilemma by splitting the noise across clients so that the aggregate is correctly calibrated while no individual party ever sees the aggregate before noise is added [103,104]. This is the clearest example in the review of a composition that is strictly better than either ordering of its components, and it is worth noting because such cases are rarer than the literature's enthusiasm for combination would suggest.

11 Choosing a Technique: A Scenario-Driven Selection Guide

A reader who has followed the preceding sections possesses a map of the field but not necessarily a decision procedure. This section supplies one. The selection logic is driven by four questions whose answers are usually known at the outset of a project: who holds the data, whether the party performing the computation may see it, whether the result is published to parties who should not learn about individuals, and what the deployment's binding resource constraint is. Fig. 5 renders this logic as a decision flow, and Table 9 maps common scenarios onto the resulting recommendations.

Three points about this procedure deserve emphasis, because a flowchart invites more confidence than the underlying evidence supports. First, the recommendations identify a starting point, not a finished architecture; every branch terminates in a combination whose compositional interactions must then be analyzed using Section 10. Second, the binding constraint is frequently organizational rather than technical. A three-party honest-majority protocol is dramatically more efficient than its dishonest-majority equivalent, but only if three mutually non-colluding parties actually exist, and in many corporate structures they do not. Third, the correct answer is sometimes to use none of these techniques: where data can be lawfully centralized and the released model is not exposed, the cost of privacy-preserving computation buys nothing, and the honest recommendation is conventional security engineering.

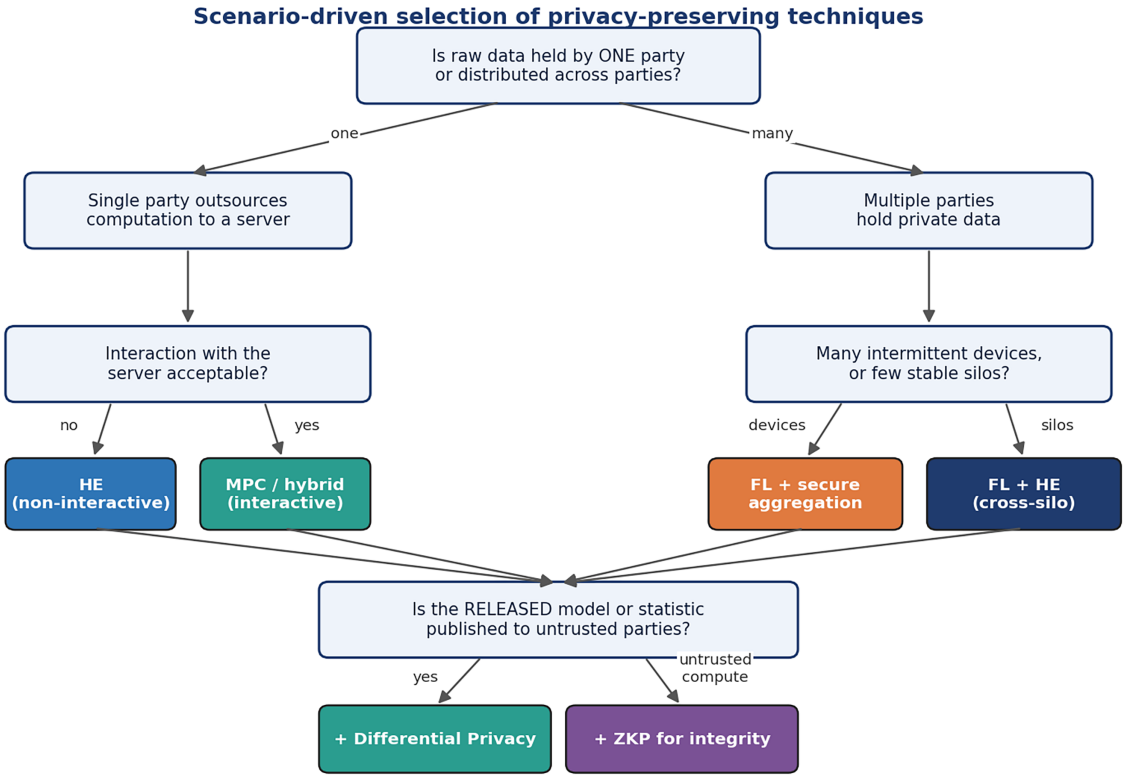


Figure 5: A scenario-driven decision flow for selecting privacy-preserving techniques. The first branch distinguishes single-party outsourcing from multi-party collaboration; the second selects between non-interactive and interactive protocols according to whether round-trip communication with the computing party is acceptable, and between device-scale and silo-scale federation according to the number and stability of participants; the final branch adds differential privacy whenever the output is made available to untrusted parties and zero-knowledge proofs whenever the computing party is untrusted for integrity as well as confidentiality. The flow yields a starting point rather than a complete design: the compositional tensions of Section 10 must still be resolved for the selected combination.

Table 9: Mapping of representative deployment scenarios to recommended technique combinations. "Binding constraint" names the resource that most often determines feasibility in that scenario. Recommendations assume the semi-honest adversary model unless stated; malicious security typically adds a factor of two to ten in cost.

Scenario	Data Holder (s)	Recommended Combination	Binding Constraint
Cloud inference on private client data	Single client, untrusted server	HE (CKKS) inference; add ZKP if server integrity is doubted	Server computation time
Hospitals jointly training a diagnostic model	Few institutions (cross-silo)	FL + homomorphic aggregation + DP on released model	Regulatory approval; silo bandwidth

(Continued)

Table 9 (continued)

Scenario	Data Holder (s)	Recommended Combination	Binding Constraint
Mobile keyboard prediction	Millions of devices (cross-device)	FL + secure aggregation + user-level DP	Client communication and dropout
Two firms computing a joint risk score	Two mutually distrustful parties	MPC (two-party, mixed protocol)	Network round trips
Publishing population statistics	Single trusted curator	Central DP with numerical accounting	Utility at target ε
Telemetry from untrusted clients	Many devices, untrusted curator	Local DP or shuffled DP	Noise-induced utility loss
Outsourced encrypted document search	Single client, untrusted storage	SSE with forward/backward privacy; ORAM if access patterns are sensitive	Leakage profile
Private fine-tuning of a language model	Single organization, sensitive corpus	DP-SGD on low-rank adapters over a public backbone	Utility at target ε
Long-horizon confidential archives	Any	Post-quantum primitives throughout; hybrid during migration	Key and signature sizes

Note: Scenarios are archetypes; real deployments frequently combine two or more, in which case the more restrictive requirement governs.

12 From Research Prototypes to Production Deployment

The performance figures collected in this review are drawn overwhelmingly from research prototypes evaluated under controlled conditions, and the distance between such a prototype and a production system is larger in this field than in most other fields. This section addresses that distance directly, because a review that discusses practicality without examining it invites the reasonable objection that its notion of practicality is untested. Fig. 6 summarizes our qualitative assessment of deployment readiness across the techniques surveyed.

12.1 Engineering Complexity and Integration with Machine-Learning Pipelines

The first obstacle is that privacy-preserving components do not compose cleanly with the tooling that surrounds modern machine learning. A production pipeline assumes it can inspect intermediate tensors, log metrics, checkpoint and resume, tune hyperparameters against a validation set, monitor for distribution drift, and debug a failing example end-to-end. Encryption defeats inspection, differential privacy makes every diagnostic query a budget expenditure, and multi-party computation turns a single-process training loop into a distributed protocol with its own failure modes. Hyperparameter tuning is particularly awkward: the search itself consumes privacy budget, and a search conducted on public proxy data may select settings that are wrong for the private task [67]. Frameworks that expose a familiar tensor interface over a cryptographic

backend, or that compile ordinary programs to protocols, exist precisely to reduce this friction [38,58,92], but they narrow rather than close the gap, and the residual engineering effort is routinely underestimated in research papers whose contribution is measured in latency.

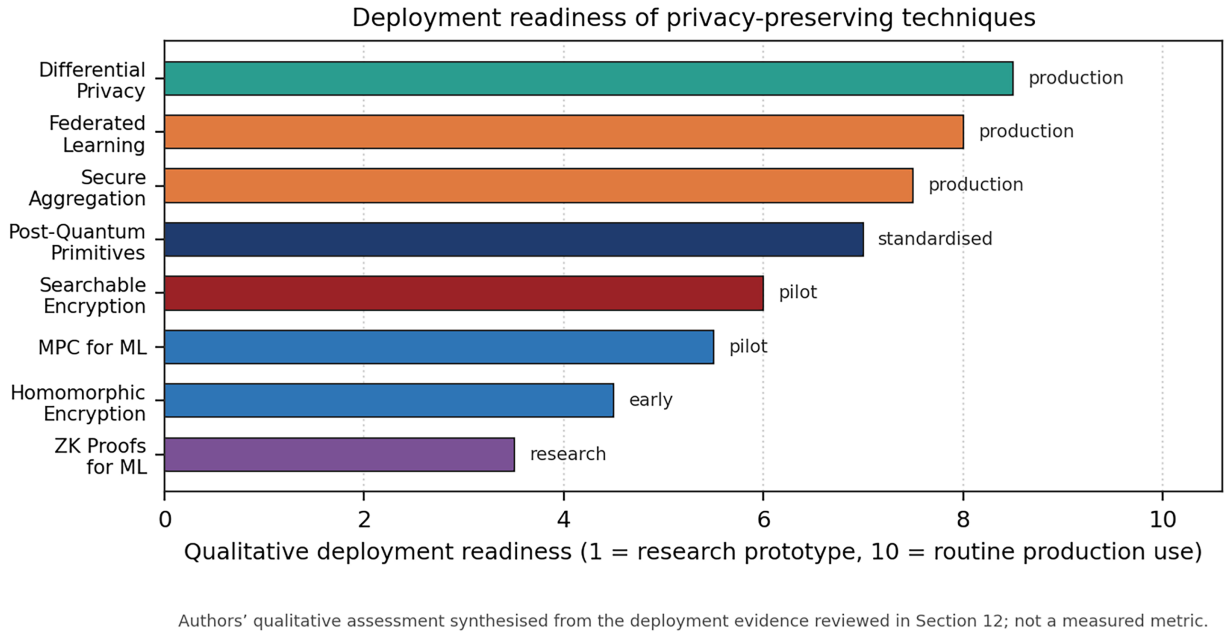


Figure 6: Qualitative deployment readiness of the privacy-preserving techniques surveyed, ordered from routine production use to research-prototype status. Positions synthesize the deployment evidence discussed in this section: differential privacy and federated learning have documented large-scale industrial and clinical deployments; secure aggregation is deployed within those federated systems; post-quantum primitives are standardized with migration under way; searchable encryption and MPC appear in pilot and niche production settings; homomorphic encryption is deployed for narrow high-value workloads; and zero-knowledge proofs for machine learning remain largely experimental. The scale is a qualitative synthesis rather than a measured metric.

12.2 Key Management and Operational Security

The second obstacle receives remarkably little attention in the research literature relative to its practical weight. Every cryptographic technique surveyed here presupposes a key-management arrangement, and the security of the deployment is bounded by that arrangement rather than by the cryptography. Homomorphic encryption requires that the secret key live somewhere the computing party cannot reach, which is straightforward when a single client both encrypts and decrypts and considerably harder when results must be shared among several parties, motivating threshold and multi-key variants in which no single participant can decrypt [100]. Federated systems must manage per-client keys at a scale of millions, handle rotation, and remain functional when clients disappear permanently. Evaluation keys for homomorphic schemes are large, must be distributed to the computing party, and become a liability if compromised. None of these problems is novel to privacy-preserving machine learning, but their combination at scale is, and a deployment whose key custody is weak derives little benefit from a scheme whose security proof is strong.

12.3 Regulatory Compliance and the Legal Status of Guarantees

The third obstacle is that formal privacy guarantees and legal privacy obligations are defined in different vocabularies and do not map onto one another cleanly. Data-protection regimes reason in terms of personal data, identifiability, purpose limitation, lawful basis and data-subject rights; differential privacy reasons

in terms of a budget parameter whose relationship to identifiability is quantitative and continuous rather than categorical. Whether a differentially private release constitutes anonymized data, and at what ε, is a question regulators have largely not answered, so organizations deploying these techniques operate without the certainty that would justify the investment [8]. Federated learning is often adopted specifically because keeping data within a jurisdiction simplifies the legal analysis, and that governance benefit, rather than any cryptographic property, is frequently the decisive argument in regulated sectors [12]. The right to erasure poses a further and largely unresolved difficulty, since removing a record from a trained model is not achievable by deleting it from the training set, and the interaction between machine unlearning and differential privacy remains an open area.

12.4 Evidence from Deployed Systems and Vertical Domains

Against these obstacles it is worth recording where deployment has genuinely occurred, since the picture is not uniformly discouraging. Local differential privacy has been used at industrial scale for telemetry collection for over a decade [69], and central differential privacy is applied to major statistical releases. Federated learning with secure aggregation operates in production on consumer mobile platforms [95,96]. The most instructive vertical is healthcare, where federated learning has progressed from feasibility studies to multi-institutional clinical collaborations: consortium projects now train models across hospitals in several countries, and dedicated open frameworks provide the auditability and governance features that clinical deployment requires [12,134]. Across the clinical literature we reviewed, a consistent pattern emerges in which the technical protocol is rarely the limiting factor: data harmonization across institutions, ethical approval, and non-identical data distributions across sites are the difficulties most frequently reported [12,135].

A second vertical worth examining is the instrumented city, where federated learning has been proposed as the privacy-preserving substrate for digital twins that mirror physical infrastructure using data from vehicles, sensors and mobile devices. Here the privacy stakes are unusual because the data is continuous, spatially indexed and often trivially re-identifiable from movement patterns alone, and the participating devices are numerous, heterogeneous and intermittently connected; surveys of this setting identify communication efficiency and device heterogeneity as the binding constraints [13]. To this we would add an observation of our own: the privacy analysis in such deployments must contend with the correlation between successive observations from the same device, which sits uneasily with the independence assumptions underlying much of the composition analysis of Section 5. Finance presents a third pattern, in which cross-silo federations among a small number of well-resourced and mutually distrustful institutions make the heavier cryptographic machinery of homomorphic aggregation economically viable [99,101]. Table 10 summarizes these domain patterns.

Table 10: Deployment patterns across vertical domains. "Typical scale" indicates the number of participating entities; "dominant obstacle" records, on our reading of the sources reviewed, the factor that most often limits deployment in practice, which is frequently organizational rather than cryptographic.

Domain	Typical Scale	Techniques in Practice	Dominant Obstacle
Healthcare/clinical research	2–50 institutions (cross-silo)	FL, secure aggregation, DP, emerging HE	Data harmonization, ethics approval, non-IID sites

(Continued)

Table 10 (continued)

Domain	Typical Scale	Techniques in Practice	Dominant Obstacle
Consumer mobile platforms	10^6–10^9 devices (cross-device)	FL, secure aggregation, user-level and local DP	Communication, dropout, device heterogeneity
Financial services	2–20 institutions	FL with homomorphic aggregation, MPC	Regulatory approval, competitive sensitivity
Smart cities and digital twins	10^4–10^7 heterogeneous devices	FL, local DP	Correlated streams, re-identification from mobility
Official statistics	Single curator	Central DP with numerical accounting	Utility at defensible ε; public communication
Cloud ML services	Single client and server	HE inference; TEEs	Latency and cost per query

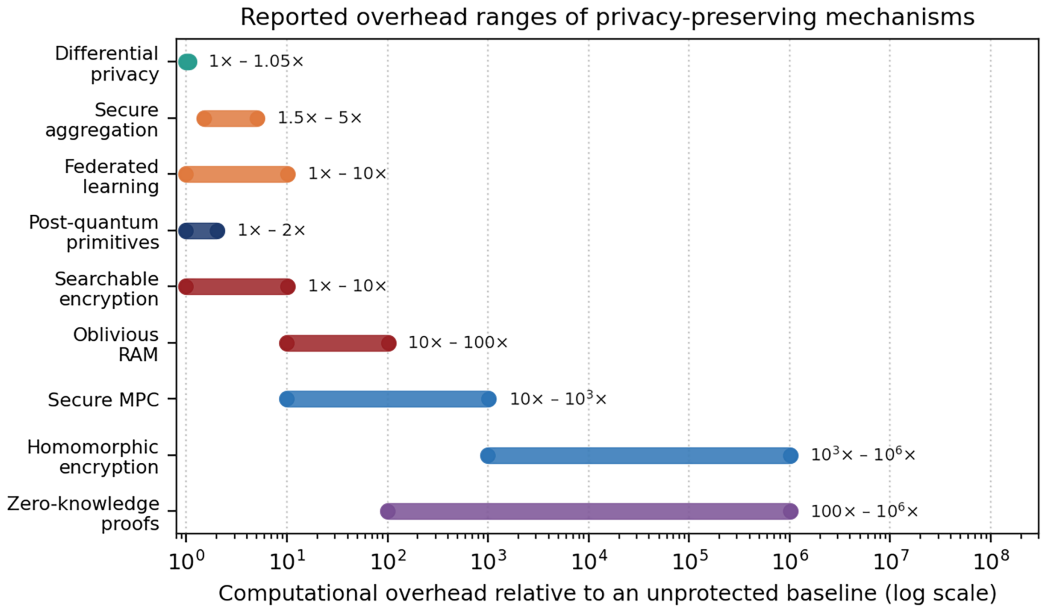
Note: Entries describe the prevailing pattern reported in the surveyed literature and are not exhaustive; several domains additionally run pilots of techniques not listed.

13 Benchmarking and Evaluation Methodology

A recurring difficulty in preparing this review, and a limitation that any reader should carry away from it, is that the quantitative results reported across the literature are not commensurable. Fig. 2 places six inference systems on a common axis while cautioning that they differ in hardware, model, dataset, batch size, threat model and security parameter; the caution is necessary, but it also concedes that the comparison it accompanies cannot bear much weight. This section states the problem precisely and describes what would be required to resolve it.

Four sources of incomparability recur. The first is threat-model variation: a system secure against a malicious adversary is doing strictly more work than one secure only against a semi-honest adversary, and reported latencies rarely make the comparison explicit. The second is security-parameter variation, since a homomorphic scheme evaluated at a smaller ring dimension is faster and less secure, and parameter selection interacts with the depth of the specific circuit. The third is the throughput-vs.-latency distinction described in Section 3.3, which can move a reported figure by three orders of magnitude in either direction depending on the batching regime. The fourth, and the most difficult to correct, is that different techniques protect different assets, so that a differentially private system and a homomorphically encrypted one cannot be compared on a single axis at all: one costs accuracy and the other costs time, and no exchange rate between them is defensible in general.

What would a unified benchmark require? At minimum, a fixed set of reference tasks spanning the regimes that matter (batch inference, interactive inference, cross-silo training, cross-device training, and analytics); a common hardware baseline with a defined scaling methodology; mandatory disclosure of threat model, corruption threshold, security parameter and, for differentially private systems, the accountant used and the resulting (ε, δ) at a stated δ; and reporting of both a utility metric and a cost metric so that points on the privacy–utility–cost surface can be located rather than inferred. Elements of this discipline already exist in parts of the field: the homomorphic-encryption community has converged on standardized parameter sets and shared security estimators [7,38], the differential-privacy community on shared accounting libraries and, more recently, on empirical auditing as a validation step [67,87], and the multi-party computation community on common frameworks that make protocol substitution straightforward [58,92]. What is absent is a cross-paradigm suite that would let a practitioner evaluate an HE, an MPC and a DP solution to the same problem on equal terms. Fig. 7 renders the resulting overhead ranges visually, and Table 11 assembles the comparison that is currently possible; the qualifications attached to it are themselves the argument for building something better.



Ranges synthesised from the primary sources cited in Sections 3–9 and summarised in Table 9; orders of magnitude only — see Section 13 on comparability.

Figure 7: Reported computational overhead of privacy-preserving mechanisms relative to an unprotected baseline, on a logarithmic scale spanning roughly eight orders of magnitude. Each bar shows the range synthesized from the primary sources cited in Sections 3–9 and tabulated in Table 11. Differential privacy is effectively free computationally and costs accuracy instead; secure aggregation and federated learning add modest compute but substantial communication; searchable encryption and oblivious RAM occupy an intermediate band; secure multi-party computation, homomorphic encryption and zero-knowledge proving are the most expensive, with proving cost depending heavily on the statement proved. Ranges are order-of-magnitude indications only, for the reasons set out in Section 13.

Table 11: Unified qualitative comparison of privacy-preserving mechanisms across the dimensions that determine deployment feasibility. Overhead ranges are order-of-magnitude summaries of the figures reported in Sections 3–9 relative to an unprotected baseline, and are indicative only: actual values depend on threat model, security parameter, model architecture and hardware, as discussed in this section. "Utility loss" refers to degradation of model accuracy attributable to the mechanism itself.

Mechanism	Asset Protected	Compute Overhead	Communication Overhead	Utility Loss	Maturity
Homomorphic encryption	Input confidentiality	10^3–$10^6\times$	Low (1–10× ciphertext expansion)	None to small (approximation)	Early production
Secure multi-party computation	Input confidentiality	10–$10^3\times$	High (grows with circuit depth)	None (exact)	Pilot
Differential privacy	Output privacy	Negligible	None	Moderate to large (depends on ε)	Production
Federated learning	Data locality	1–10×	High (per-round model transfer)	Small to moderate (non-IID)	Production
Secure aggregation	Individual update confidentiality	Low	Moderate (protocol-dependent)	None	Production
Searchable encryption	Query and data confidentiality (with leakage)	1–10×	Low	None	Pilot
Oblivious RAM	Access-pattern confidentiality	10–100×	Logarithmic in database size	None	Niche
Zero-knowledge proofs	Integrity of computation	10^2–$10^6\times$ (proving)	Low (succinct proofs)	None	Research
Post-quantum primitives	Long-term confidentiality	1–2×	Low (larger keys and signatures)	None	Standardized

Note: Ranges are synthesized from the primary sources cited in the corresponding sections and are intended to support technique selection, not to substitute for measurement on a specific workload.

Because this area moves quickly, we note the systematizations that appeared as this review was being completed and that a reader should consult for developments beyond our cut-off. Private inference for large language models under approximate homomorphic encryption now has a dedicated systematization covering packing layouts and model-preservation strategies for non-interactive CKKS-based inference [136], and the multi-party side has a companion systematization unifying secret-sharing schemes across two-, three- and four-party settings together with the conversions between them [94]. Both postdate most of the

primary sources surveyed above and together indicate the direction of travel: toward large-model workloads and toward hybrid protocols that move fluidly between representations [23,53].

14 Cross-Cutting Challenges and Future Directions

Reading across the seven areas, several challenges recur and, in our assessment, define the research agenda for the coming years. We organize them around the cost dimensions introduced in Section 2.

The first is computational throughput for homomorphic encryption. Despite hundredfold accelerations from GPUs and continuing algorithmic improvements to bootstrapping, encrypted inference for large models remains orders of magnitude slower than plaintext, and encrypted training is largely out of reach [36,49,56]. Progress will come from co-design across the stack: hardware accelerators tailored to the number-theoretic transform, scheme parameters chosen jointly with model architecture, and model designs that minimize multiplicative depth and replace HE-hostile non-linearities with friendly approximations [53,55]. The transformer-inference systems of Sections 3 and 6 are early exemplars of this co-design philosophy, and we expect it to deepen.

The second is tight privacy accounting and its honest reporting. Numerical accountants have made it possible to compute the privacy loss of practical mechanisms precisely, narrowing the gap between provable and intuitive privacy [77,78,80]. Yet critical reviews continue to find that deployed systems report privacy budgets too large to be meaningful, or that subtle accounting and implementation errors void the intended guarantee [8,67]. The open problem is partly technical—extending tight accounting to the hidden-state setting where only the final model is released, and to mechanisms beyond the subsampled Gaussian—and partly methodological, requiring standardized auditing and reporting so that a stated ε means the same thing across papers and products.

The third is communication efficiency for interactive protocols. Multi-party computation and secure aggregation convert privacy into network traffic, and for cross-device federated learning over millions of intermittently connected clients this is the binding constraint [97,98]. Advances in sparsification, quantization, and one-shot mask reconstruction have helped, but the fundamental tension between dropout resilience, malicious security, and communication remains unresolved at the largest scales [99,103].

The fourth is leakage quantification. Searchable encryption and oblivious RAM trade precisely characterized leakage for efficiency, and the steady stream of leakage-abuse and injection attacks demonstrates that informal leakage arguments are unreliable [118,119]. A principled, composable theory of leakage—one that lets a system designer reason about the cumulative leakage of a pipeline that combines SSE, ORAM, and machine learning, and that connects leakage to a differential-privacy-style budget—would be a significant advance [24,25].

The fifth is the post-quantum transition. With standards now published, the work shifts to migration: auditing existing privacy-preserving systems for quantum-vulnerable components, integrating ML-KEM and ML-DSA into the protocols that surround private computation, and absorbing the performance impact of larger keys and signatures [126,128,132]. Because much of the PPML stack already rests on lattice assumptions, the field is better positioned than most, but the integration work is substantial and largely undone.

Cutting across all five is the central theme of this review: composition. The strongest privacy-preserving systems are not built from a single technique but from a careful composition of several, arranged so that the residual leakage of one is covered by the guarantee of another—homomorphic aggregation under a federated protocol whose output is sanitized by differential privacy and whose transport is secured by post-quantum

key exchange [21,22,104]. Reasoning rigorously about such compositions, rather than about each technique in isolation, is in our view the defining methodological challenge of the field.

A sixth challenge, orthogonal to the five cost-oriented ones, is standardization and benchmarking, analyzed in detail in Section 13. The quantitative results collected throughout this review are difficult to compare precisely because authors measure different models on different hardware under different threat models, and the absence of common benchmarks impedes cumulative progress [1,7]. The homomorphic-encryption community has begun to converge on standardized parameter sets and security estimators, and the differential-privacy community on shared accounting libraries and empirical auditing protocols [87], but a unified, cross-paradigm benchmark suite—one that would let a practitioner compare an HE, an MPC, and a DP solution to the same task on equal footing—does not yet exist. Establishing such benchmarks, together with usable tooling that hides cryptographic parameter selection behind safe defaults, is a prerequisite for the field's transition from research artifacts to dependable infrastructure [38,67].

Finally, the regulatory and societal context increasingly shapes technical priorities. Data-protection regimes that mandate purpose limitation, data minimization, and the right to erasure interact non-trivially with machine learning: erasure, for instance, raises the question of machine unlearning, whose interaction with differential privacy and with the immutability of trained models is only beginning to be understood [8,21]. As privacy-preserving techniques move from the laboratory into regulated industries, the criteria for success will broaden from raw performance to include auditability, interpretability of the privacy guarantee to non-experts, and resilience to the evolving threat landscape—including the quantum adversary whose eventual arrival the post-quantum standards anticipate [20,132].

15 Conclusion

Privacy-preserving machine learning has progressed from a collection of impractical curiosities to a maturing engineering discipline with a coherent set of tools and a growing record of deployment. Homomorphic encryption now supports deep-network and transformer inference at costs that, while high, are no longer prohibitive for high-value applications; differential privacy provides a rigorous and composable language for output privacy, backed by accountants that compute privacy loss precisely; secure multi-party computation and federated learning enable collaborative learning across organizational and device boundaries; encrypted search and oblivious storage protect the data layer; and the 2024 post-quantum standards ensure that these guarantees will endure. No single technique suffices, and the central lesson of the literature is that real systems must compose them, matching each technique to the threat model and resource budget it best serves. The open problems—bootstrapping throughput, tight and honest privacy accounting, communication efficiency, leakage quantification, and quantum-resilient migration—are concrete and tractable, and progress on each will broaden the range of applications for which strong privacy and useful machine learning can be achieved simultaneously. We hope this review serves both as an orientation for those entering the field and as a structured reference for those advancing it.

Looking ahead, the most consequential progress is likely to come not from any single technique maturing in isolation but from principled composition: homomorphic encryption accelerated by purpose-built hardware and enriched by functional bootstrapping, differential privacy with tight and empirically audited accounting, federated and multi-party protocols whose communication has been driven down to practical levels, zero-knowledge proofs supplying the integrity that confidentiality mechanisms remove, storage layers whose leakage is quantified rather than hoped away, and a uniformly post-quantum base. Sections 10–13 argue that such composition is harder than the customary appeal to complementarity suggests: techniques conflict as well as combine, the choice among them is driven as much by organizational structure as by cryptography, and the distance from prototype to production is dominated by key management, tooling

and regulatory uncertainty rather than protocol efficiency. The open problems that follow are concrete: bootstrapping throughput, tight and auditable accounting, communication efficiency, leakage quantification, unified benchmarking, and quantum-resilient migration.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Quang-Vinh Dang, Dat Le; literature search and screening: Quang-Vinh Dang, Minh Ngoc Dinh, Ngoc-Son-An Nguyen; analysis and interpretation of the surveyed results: Quang-Vinh Dang, Dat Le, Minh Ngoc Dinh; design of the taxonomy, compositional analysis and selection guide: Quang-Vinh Dang, Dat Le; preparation of figures and tables: Minh Ngoc Dinh, Ngoc-Son-An Nguyen; draft manuscript preparation: Quang-Vinh Dang, Dat Le; critical revision for important intellectual content: Dat Le, Minh Ngoc Dinh, Ngoc-Son-An Nguyen. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable; this is a review article and no new data were generated or analyzed.

Ethics Approval: Not applicable.

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

References

1. Zhou I, Tofigh F, Piccardi M, Abolhasan M, Franklin D, Lipman J. Secure multi-party computation for machine learning: a survey. IEEE Access. 2024;12:53881–99. doi:10.1109/ACCESS.2024.3388992.
2. El Ouadrhiri A, Abdelhadi A. Differential privacy for deep and federated learning: a survey. IEEE Access. 2022;10:22359–80. doi:10.1109/ACCESS.2022.3151670.
3. Shokri R, Stronati M, Song C, Shmatikov V. Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP); 2017 May 22–26; San Jose, CA, USA. p. 3–18. doi:10.1109/SP.2017.41.
4. Carlini N, Chien S, Nasr M, Song S, Terzis A, Tramèr F. Membership inference attacks from first principles. In: 2022 IEEE Symposium on Security and Privacy (SP); 2022 May 22–26; San Francisco, CA, USA. p. 1897–914. doi:10.1109/SP46214.2022.9833649.
5. Zhu L, Liu Z, Han S. Deep leakage from gradients [Internet]. 2019 [cited 2026 Jan 1]. Available from: https://proceedings.neurips.cc/paper/2019/hash/60a6c4002cc7b29142def8871531281a-Abstract.html.
6. Geiping J, Bauermeister H, Dröge H, Moeller M. Inverting gradients-how easy is it to break privacy in federated learning? [Internet]. 2020 [cited 2026 Jan 1]. Available from: https://proceedings.neurips.cc/paper/2020/hash/c4ede56bbd98819ae6112b20ac6bf145-Abstract.html.
7. Marcolla C, Sucasas V, Manzano M, Bassoli R, Fitzek FHP, Aaraj N. Survey on fully homomorphic encryption, theory, and applications. Proc IEEE. 2022;110(10):1572–609. doi:10.1109/JPROC.2022.3205665.
8. Blanco-Justicia A, Sánchez D, Domingo-Ferrer J, Muralidhar K. A critical review on the use (and misuse) of differential privacy in machine learning. ACM Comput Surv. 2023;55(8):1–16. doi:10.1145/3547139.
9. Feng D, Yang K. Concretely efficient secure multi-party computation protocols: survey and more. Secur Saf. 2022;1:2021001. doi:10.1051/sands/2021001.
10. Liu T, Xie X, Zhang Y. zkCNN: zero knowledge proofs for convolutional neural network predictions and accuracy. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security; 2021 Nov 15–19; Virtual Event. p. 2968–85. doi:10.1145/3460120.3485379.
11. Garg S, Goel A, Jha S, Mahloujifar S, Mahmoody M, Policharla GV, et al. Experimenting with zero-knowledge proofs of training. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: ACM; 2023. p. 1880–94. doi:10.1145/3576915.3623202.

12. Bakas S, Li X, Shah P, Roth HR. Federated learning in healthcare: from research to real-world deployment. Annu Rev Biomed Eng. 2026;28(1):163–86. doi:10.1146/annurev-bioeng-080125-041414.
13. Ramu SP, Boopalan P, Pham QV, Maddikunta PKR, Huynh-The T, Alazab M, et al. Federated learning enabled digital twins for smart cities: concepts, recent advances, and future directions. Sustain Cities Soc. 2022;79:103663. doi:10.1016/j.scs.2021.103663.
14. Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, et al. Advances and open problems in federated learning. Found Trends® Mach Learn. 2021;14(1–2):1–210. doi:10.1561/2200000083.
15. Mohassel P, Rindal P. ABY^3: a mixed protocol framework for machine learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: ACM; 2018. p. 35–52. doi:10.1145/3243734.3243760.
16. Mohassel P, Zhang Y. SecureML: a system for scalable privacy-preserving machine learning. In: Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP); 2017 May 22–26; San Jose, CA, USA. p. 19–38. doi:10.1109/SP.2017.12.
17. Damgård I, Pastro V, Smart N, Zakarias S. Multiparty computation from somewhat homomorphic encryption. In: Advances in cryptology—CRYPTO 2012. Berlin/Heidelberg, Germany: Springer; 2012. p. 643–62. doi:10.1007/978-3-642-32009-5_38.
18. Dwork C, McSherry F, Nissim K, Smith A. Calibrating noise to sensitivity in private data analysis. In: Theory of cryptography. Berlin/Heidelberg, Germany: Springer; 2006. p. 265–84. doi:10.1007/11681878_14.
19. Dwork C, Roth A. The algorithmic foundations of differential privacy. Found Trends Theor Comput Sci. 2014;9(3–4):211–487. doi:10.1561/0400000042.
20. Shor PW. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J Comput. 1997;26(5):1484–509. doi:10.1137/s0097539795293172.
21. Fu J, Hong Y, Ling X, Wang L, Ran X, Sun Z, et al. Differentially private federated learning: a systematic review. arXiv:2405.08299. 2024.
22. Sav S, Pyrgelis A, Troncoso-Pastoriza JR, Froelicher D, Bossuat JP, Sousa JS, et al. POSEIDON: privacy-preserving federated neural network learning. In: Proceedings 2021 Network and Distributed System Security Symposium; 2021 Feb 21–25; Virtual. Internet. doi:10.14722/ndss.2021.24119.
23. Pang Q, Zhu J, Möllering H, Zheng W, Schneider T. BOLT: privacy-preserving, accurate and efficient inference for transformers. In: Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP); 2024 May 19–23; San Francisco, CA, USA. p. 4753–71. doi:10.1109/SP54263.2024.00130.
24. Li F, Ma J, Miao Y, Liu X, Ning J, Deng RH. A survey on searchable symmetric encryption. ACM Comput Surv. 2024;56(5):1–42. doi:10.1145/3617991.
25. Gui Z, Paterson KG, Patranabis S. Rethinking searchable symmetric encryption. In: Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP); 2023 May 21–25; San Francisco, CA, USA. p. 1401–18. doi:10.1109/SP46215.2023.10179460.
26. Rivest RL, Adleman L, Dertouzos ML. On data banks and privacy homomorphisms. Found Secure Comput. 1978;4(11):169–80.
27. Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) fully homomorphic encryption without bootstrapping. ACM Trans Comput Theory. 2014;6(3):1–36. doi:10.1145/2633600.
28. Fan J, Vercauteren F. Somewhat practical fully homomorphic encryption (Paper 2012/144) [Internet]. IACR Cryptol ePrint Arch; 2012 [cited 2026 Jan 1]. Available from: https://eprint.iacr.org/2012/144.
29. Gentry C, Sahai A, Waters B. Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in cryptology—CRYPTO 2013. Berlin/Heidelberg, Germany: Springer; 2013. p. 75–92. doi:10.1007/978-3-642-40041-4_5.
30. Cheon JH, Kim A, Kim M, Song Y. Homomorphic encryption for arithmetic of approximate numbers. In: Advances in cryptology—ASIACRYPT 2017. Cham, Switzerland: Springer International Publishing; 2017. p. 409–37. doi:10.1007/978-3-319-70694-8_15.
31. Chillotti I, Gama N, Georgieva M, Izabachène M. TFHE: fast fully homomorphic encryption over the torus. J Cryptol. 2020;33(1):34–91. doi:10.1007/s00145-019-09319-x.

32. Li B, Micciancio D. On the security of homomorphic encryption on approximate numbers. In: Advances in cryptology—EUROCRYPT 2021. Cham, Switzerland: Springer International Publishing; 2021. p. 648–77. doi:10.1007/978-3-030-77870-5_23.
33. Al Badawi A, Polyakov Y. Demystifying bootstrapping in fully homomorphic encryption (Paper 2023/149) [Internet]. IACR Cryptol ePrint Arch; 2023 [cited 2026 Jan 1]. Available from: https://eprint.iacr.org/2023/149.
34. Bae Y, Cheon JH, Cho W, Kim J, Kim T. META-BTS: bootstrapping precision beyond the limit. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: ACM; 2022. p. 223–34. doi:10.1145/3548606.3560696.
35. Coron JS, Seuré T. PaCo: bootstrapping for CKKS via partial CoeffToSlot. In: Advances in cryptology—ASIACRYPT 2025. Singapore: Springer Nature; 2026. p. 36–67. doi:10.1007/978-981-95-5122-4_2.
36. Yan L, Zeng P, Cao H, Song P, Wang M. Faster bootstrapping for CKKS with less modulus consumption (Paper 2025/1403) [Internet]. IACR Cryptol ePrint Arch; 2025 [cited 2026 Jan 1]. Available from: https://eprint.iacr.org/2025/1403.
37. Cheon JH, Han K, Kim A, Kim M, Song Y. Bootstrapping for approximate homomorphic encryption. In: Advances in cryptology—EUROCRYPT 2018. Cham, Switzerland: Springer International Publishing; 2018. p. 360–84. doi:10.1007/978-3-319-78381-9_14.
38. Al Badawi A, Bates J, Bergamaschi F, Cousins DB, Erabelli S, Genise N, et al. OpenFHE: open-source fully homomorphic encryption library. In: Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography; 2022 Nov 7; Los Angeles, CA, USA. p. 53–63. doi:10.1145/3560827.3563379.
39. Cheon JH, Hanrot G, Kim J, Stehlé D. SHIP: a shallow and highly parallelizable CKKS bootstrapping algorithm. In: Advances in cryptology—EUROCRYPT 2025. Cham, Switzerland: Springer Nature; 2025. p. 398–428. doi:10.1007/978-3-031-91131-6_14.
40. Coron JS, Köstler R. Low-latency bootstrapping for CKKS using roots of unity (Paper 2025/651) [Internet]. IACR Cryptol ePrint Arch; 2025 [cited 2026 Jan 1]. Available from: https://eprint.iacr.org/2025/651.
41. Alexandru A, Kim A, Polyakov Y. General functional bootstrapping using CKKS. In: Advances in cryptology—CRYPTO 2025. Cham, Switzerland: Springer Nature; 2025. p. 304–37. doi:10.1007/978-3-032-01881-6_10.
42. Bae Y, Cheon JH, Kim J, Stehlé D. Bootstrapping bits with CKKS. In: Advances in cryptology—EUROCRYPT 2024. Cham, Switzerland: Springer Nature; 2024. p. 94–123. doi:10.1007/978-3-031-58723-8_4.
43. Bae Y, Kim J, Stehlé D, Suvanto E. Bootstrapping small integers with CKKS. In: Advances in cryptology—ASIACRYPT 2024. Singapore: Springer Nature; 2025. p. 330–60. doi:10.1007/978-981-96-0875-1_11.
44. Lu WJ, Huang Z, Hong C, Ma Y, Qu H. PEGASUS: bridging polynomial and non-polynomial evaluations in homomorphic encryption. In: 2021 IEEE Symposium on Security and Privacy (SP); 2021 May 24–27; San Francisco, CA, USA. p. 1057–73. doi:10.1109/SP40001.2021.00043.
45. Gilad-Bachrach R, Dowlin N, Laine K, Lauter K, Naehrig M, Wernsing J. CryptoNets: applying neural networks to encrypted data with high throughput and accuracy. In: Proceedings of the 33rd International Conference on Machine Learning (ICML); 2016 Jun 19–24; New York, NY, USA. p. 201–10.
46. Dathathri R, Saarikivi O, Chen H, Laine K, Lauter K, Maleki S, et al. CHET: an optimizing compiler for fully-homomorphic neural-network inferencing. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation; 2019 Jun 22–26; Phoenix, AZ, USA. p. 142–56. doi:10.1145/3314221.3314628.
47. Dathathri R, Kostova B, Saarikivi O, Dai W, Laine K, Musuvathi M. EVA: an encrypted vector arithmetic language and compiler for efficient homomorphic computation. In: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation; 2020 Jun 15–20; London, UK. p. 546–61. doi:10.1145/3385412.3386023.
48. Lee JW, Kang H, Lee Y, Choi W, Eom J, Deryabin M, et al. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. IEEE Access. 2022;10:30039–54. doi:10.1109/ACCESS.2022.3159694.

49. Lee E, Lee JW, Lee J, Kim YS, Kim Y, No JS, et al. Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In: Proceedings of the 39th International Conference on Machine Learning; 2022 Jul 17–23; Baltimore, MD, USA. p. 12403–22.
50. Kim D, Guyot C. Optimized privacy-preserving CNN inference with fully homomorphic encryption. IEEE Trans Inf Forensics Secur. 2023;18:2175–87. doi:10.1109/TIFS.2023.3263631.
51. Lu WJ, Huang Z, Gu Z, Li J, Liu J, Hong C, et al. BumbleBee: secure two-party inference framework for large transformers [Internet]. 2025 [cited 2026 Jan 1]. Available from: https://eprint.iacr.org/2023/1678.
52. Zhang J, Yang X, He L, Chen K, Lu WJ, Wang Y, et al. Secure transformer inference made non-interactive. In: Proceedings 2025 Network and Distributed System Security Symposium; 2025 Feb 24–28; San Diego, CA, USA. doi:10.14722/ndss.2025.230868.
53. Moon J, Yoo D, Jiang X, Kim M. THOR: secure transformer inference with homomorphic encryption. In: Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security; 2025 Oct 13–17; Taipei Taiwan. p. 3765–79. doi:10.1145/3719027.3765150.
54. Dong Y, Lu WJ, Zheng Y, Wu H, Zhao D, Tan J, et al. *PUMA*: secure inference of LLaMA-7B in five minutes. arXiv:2307.12533. 2023. doi:10.1051/sands/2025014.
55. Rho D, Kim T, Park M, Kim JW, Chae H, Ryu EK, et al. Encryption-friendly LLM architecture. arXiv:2410.02486. 2024.
56. Jung W, Kim S, Ahn JH, Cheon JH, Lee Y. Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with GPUs. IACR Trans Cryptogr Hardw Embed Syst. 2021:114–48. doi:10.46586/tches.v2021.i4.114-148.
57. Yang H, Shen S, Dai W, Zhou L, Liu Z, Zhao Y. Phantom: a CUDA-accelerated word-wise homomorphic encryption library. IEEE Trans Dependable Secure Comput. 2024;21(5):4895–906. doi:10.1109/TDSC.2024.3363900.
58. Knott B, Venkataraman S, Hannun A, Sengupta S, Ibrahim M, van der Maaten L. CrypTen: secure multi-party computation meets machine learning [Internet]. [cited 2026 Jan 1]. Available from: https://proceedings.neurips.cc/paper/2021/hash/2754518221cfbc8d25c13a06a4cb8421-Abstract.html.
59. Zhang J, Cheng X, Yang L, Hu J, Liu X, Chen K. SoK: fully homomorphic encryption accelerators. ACM Comput Surv. 2024;56(12):1–32. doi:10.1145/3676955.
60. Pindado Z, Spendlhofer T, Allam M, Mehta P, Martens L, Peña AJ. SoK of private deep neural network inference with approximate fully homomorphic encryption (Paper 2026/047) [Internet]. IACR Cryptol ePrint Arch; 2026 [cited 2026 Jan 1]. Available from: https://eprint.iacr.org/2026/047.
61. Özcan AŞ, Savaş E. HEonGPU: a GPU-based fully homomorphic encryption library 1.0 (Paper 2024/1543) [Internet]. IACR Cryptol ePrint Arch; 2024 [cited 2026 Jan 1]. Available from: https://eprint.iacr.org/2024/1543.
62. DESILO. Liberate.FHE: a new FHE library for bridging the gap between theory and practice with a focus on performance and accuracy. 2023 [cited 2026 Jan 1]. Available from: https://github.com/Desilo/liberate-fhe.
63. Özcan AŞ., Ayduman C, Türkoğlu ER, Savaş E. Homomorphic encryption on GPU. IEEE Access. 2023;11:84168–86. doi:10.1109/ACCESS.2023.3265583.
64. Lin C, Yang K, Xu T, Liang L, Wang Y, Chen Z, et al. Ironman: accelerating oblivious transfer extension for privacy-preserving AI with near-memory processing. arXiv:2507.16391. 2025.
65. Zhang B, Lu G, Qiu P, Gui X, Shi Y. Advancing federated learning through verifiable computations and homomorphic encryption. Entropy. 2023;25(11):1550. doi:10.3390/e25111550.
66. Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, et al. Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security; 2016 Oct 24–28; Vienna, Austria. p. 308–18. doi:10.1145/2976749.2978318.
67. Ponomareva N, Hazimeh H, Kurakin A, Xu Z, Denison C, McMahan HB, et al. How to DP-fy ML: a practical guide to machine learning with differential privacy. J Artif Intell Res. 2023;77:1113–201. doi:10.1613/jair.1.14649.
68. Pan K, Ong YS, Gong M, Li H, Qin AK, Gao Y. Differential privacy in deep learning: a literature survey. Neurocomputing. 2024;589:127663. doi:10.1016/j.neucom.2024.127663.

69. Erlingsson Ú., Pihur V, Korolova A. RAPPOR: randomized aggregatable privacy-preserving ordinal response. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security; 2014 Nov 3–7; Scottsdale, AZ, USA. p. 1054–67. doi:10.1145/2660267.2660348.
70. Li X, Tramèr F, Liang P, Hashimoto T. Large language models can be strong differentially private learners. In: Proceedings of the 10th International Conference on Learning Representations (ICLR); 2022 Apr 25–29; Online.
71. Liu XY, Zhu R, Zha D, Gao J, Zhong S, White M, et al. Differentially private low-rank adaptation of large language model using federated learning. ACM Trans Manage Inf Syst. 2025;16(2):1–24. doi:10.1145/3682068.
72. Xu J, Saravanan K, van Dalen R, Mehmood H, Tuckey D, Ozay M. DP-DyLoRA: fine-tuning transformer-based models on-device under differentially private federated learning using dynamic low-rank adaptation. arXiv:2405.06368. 2024.
73. Wu F, Li Z, Li Y, Ding B, Gao J. FedBiOT: LLM local fine-tuning in federated learning without full model. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining; 2024 Aug 25–29; Barcelona, Spain. p. 3345–55. doi:10.1145/3637528.3671897.
74. McKenna R, Huang Y, Sinha A, Balle B, Charles Z, Choquette-Choo CA, et al. Scaling laws for differentially private language models. arXiv:2501.18914. 2025.
75. Dwork C, Rothblum GN, Vadhan S. Boosting and differential privacy. In: 2010 IEEE 51st Annual Symposium on Foundations of Computer Science; 2010 Oct 23–26; Las Vegas, NV, USA. p. 51–60. doi:10.1109/FOCS.2010.12.
76. Kairouz P, Oh S, Viswanath P. The composition theorem for differential privacy. IEEE Trans Inf Theory. 2017;63(6):4037–49. doi:10.1109/TIT.2017.2685505.
77. Gopi S, Lee YT, Wutschitz L. Numerical composition of differential privacy [Internet]. [cited 2026 Jan 1]. Available from: https://proceedings.neurips.cc/paper/2021/hash/6097d8f3714205740f30debe1166744e-Abstract.html.
78. Doroshenko V, Ghazi B, Kamath P, Kumar R, Manurangsi P. Connect the dots: tighter discrete approximations of privacy loss distributions. Proc Priv Enhancing Technol. 2022;2022(4):552–70. doi:10.56553/popets-2022-0122.
79. Ghazi B, Kamath P, Kumar R, Manurangsi P. Faster privacy accounting via evolving discretization. In: Proceedings of the 39th International Conference on Machine Learning; 2022 Jul 17–23; Baltimore, MD, USA.
80. Wang JT, Mahloujifar S, Wu T, Jia R, Mittal P. A randomized approach to tight privacy accounting. In: Advances in Neural Information Processing Systems 36 (NeurIPS 2023); 2023 Dec 10–16; New Orleans, LA, USA.
81. Erlingsson Ú, Feldman V, Mironov I, Raghunathan A, Talwar K, Thakurta A. Amplification by shuffling: from local to central differential privacy via anonymity. In: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics; 2019. p. 2468–79. doi:10.1137/1.9781611975482.151.
82. Balle B, Bell J, Gascón A, Nissim K. The privacy blanket of the shuffle model. In: Advances in cryptology—CRYPTO 2019. Cham, Switzerland: Springer International Publishing; 2019. p. 638–67. doi:10.1007/978-3-030-26951-7_22.
83. Feldman V, McMillan A, Talwar K. Hiding among the clones: a simple and nearly optimal analysis of privacy amplification by shuffling. In: 2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS); 2022 Feb 7–10; Denver, CO, USA. p. 954–64. doi:10.1109/FOCS52979.2021.00096.
84. Mironov I. Rényi differential privacy. In: Proceedings of the 2017 IEEE 30th Computer Security Foundations Symposium (CSF); 2017 Aug 21–25; Santa Barbara, CA, USA. p. 263–75. doi:10.1109/CSF.2017.11.
85. Dong J, Roth A, Su WJ. Gaussian differential privacy. J R Stat Soc Ser B Stat Methodol. 2022;84(1):3–37. doi:10.1111/rssb.12454.
86. Nasr M, Hayes J, Steinke T, Balle B, Tramèr F, Jagielski M, et al. Tight auditing of differentially private machine learning. In: Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23); 2023 Aug 8–11; Anaheim, CA, USA. p. 1631–48.
87. Steinke T, Nasr M, Jagielski M. Privacy auditing with one (1) training run [Internet]. [cited 2026 Jan 1]. Available from: https://openreview.net/forum?id=q15zG9CHi8.
88. Mahloujifar S, Melis L, Chaudhuri K. Auditing f-differential privacy in one run. In: Proceedings of the 42nd International Conference on Machine Learning (ICML); 2025 Jul 13–19; Vancouver, BC, Canada.
89. Yao AC. Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science (SFCS 1982); 1982 Nov 3–5; Chicago, IL, USA. p. 160–4. doi:10.1109/SFCS.1982.38.

90. Watson JL, Wagh S, Popa RA. Piranha: a GPU platform for secure computation. In: Proceedings of the 31st USENIX Security Symposium; 2022 Aug 10–12; Boston, MA, USA. p. 827–44.
91. Harth-Kitzerow C, Wang Y, Rajat R, Carle G, Annavaram M. PIGEON: a high throughput framework for private inference of neural networks using secure multiparty computation. Proc Priv Enhancing Technol. 2025;2025(3):88–105. doi:10.56553/popets-2025-0090.
92. Ma J, Zheng Y, Feng J, Zhao D, Wu H, Fang W, et al. SecretFlow-SPU: a performant and user-friendly framework for privacy-preserving machine learning. In: 2023 USENIX Annual Technical Conference (USENIX ATC 23); 2023 Jul 10–12; Boston, MA, USA. p. 17–33.
93. Li D, Shao R, Wang H, Guo H, Xing EP, Zhang H. MPCFormer: fast, performant and private transformer inference with MPC. arXiv:2211.01452. 2022.
94. Zbudila M, Suresh A, Yalame H, Mirzamohammadi O, Abidin A, Preneel B. SoK: connecting the dots in privacy-preserving ML—systematization of MPC protocols and conversions between secret sharing schemes (Paper 2025/1679) [Internet]. IACR Cryptol ePrint Arch. 2025 [cited 2026 Jan 1]. Available from: https://eprint.iacr.org/2025/1679.
95. McMahan HB, Moore E, Ramage D, Hampson S, Agüera y Arcas B. Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS); 2017 Apr 20–22; Fort Lauderdale, FL, USA.
96. Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, et al. Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security; 2017 Oct 30–Nov 3; Dallas, TX, USA. p. 1175–91. doi:10.1145/3133956.3133982.
97. Bell JH, Bonawitz KA, Gascón A, Lepoint T, Raykova M. Secure single-server aggregation with (poly)logarithmic overhead. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security; 2022 Noc 9–13; Online. p. 1253–69. doi:10.1145/3372297.3417885.
98. So J, He C, Yang CS, Li S, Yu Q, Ali RE, et al. LightSecAgg: a lightweight and versatile design for secure aggregation in federated learning. In: Proceedings of Machine Learning and Systems; 2022 Aug 29–Sep 1; Santa Clara, CA, USA. p. 694–720.
99. Zhang C, Li S, Xia J, Wang W, Yan F, Liu Y. BatchCrypt: efficient homomorphic encryption for cross-*Silo* federated learning. In: Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC 20); 2020 Jul 15–17; Online. p. 493–506.
100. Ma J, Naas SA, Sigg S, Lyu X. Privacy-preserving federated learning based on multi-key homomorphic encryption. Int J Intelligent Sys. 2022;37(9):5880–901. doi:10.1002/int.22818.
101. Jin W, Yao Y, Han S, Gu J, Joe-Wong C, Ravi S, et al. FedML-HE: an efficient homomorphic-encryption-based privacy-preserving federated learning system. arXiv:2303.10837. 2023.
102. Rigaki M, Garcia S. A survey of privacy attacks in machine learning. ACM Comput Surv. 2024;56(4):1–34. doi:10.1145/3624010.
103. Li Z, Zhao H, Li B, Chi Y. SoteriaFL: a unified framework for private federated learning with communication compression [Internet]. [cited 2026 Jan 1]. Available from: https://proceedings.neurips.cc/paper_files/paper/2022/hash/1b645a77cf48821afc3ee7e5b5d42617-Abstract-Conference.html.
104. Kairouz P, Liu Z, Steinke T. The distributed discrete Gaussian mechanism for federated learning with secure aggregation. In: Proceedings of the 38th International Conference on Machine Learning (ICML); 2021 Jul 18–24; Online.
105. Wei K, Li J, Ding M, Ma C, Su H, Zhang B, et al. User-level privacy-preserving federated learning: analysis and performance optimization. IEEE Trans Mob Comput. 2022;21(9):3388–401. doi:10.1109/TMC.2021.3056991.
106. Xie Q, Jiang S, Jiang L, Huang Y, Zhao Z, Khan S, et al. Efficiency optimization techniques in privacy-preserving federated learning with homomorphic encryption: a brief survey. IEEE Internet Things J. 2024;11(14):24569–80. doi:10.1109/JIOT.2024.3382875.
107. Blanchard P, El Mhamdi EM, Guerraoui R, Stainer J. Machine learning with adversaries: Byzantine tolerant gradient descent. In: Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017); 2017 Dec 4–9; Long Beach, CA, USA. p. 119–29.

108. Yin D, Chen Y, Kannan R, Bartlett P. Byzantine-robust distributed learning: towards optimal statistical rates. In: Proceedings of the 35th International Conference on Machine Learning (ICML); 2018 Jul 10–15; Stockholm, Sweden. p. 5650–9.
109. Cao X, Fang M, Liu J, Gong NZ. FLTrust: Byzantine-robust federated learning via trust bootstrapping. In: Proceedings 2021 Network and Distributed System Security Symposium; 2021 Feb 21–25; Virtual. doi:10.14722/ndss.2021.24434.
110. Xiang Z, Wang T, Lin W, Wang D. Practical differentially private and Byzantine-resilient federated learning. arXiv:2304.09762. 2023. doi:10.1145/3589264.
111. Tan AZ, Yu H, Cui L, Yang Q. Towards personalized federated learning. IEEE Trans Neural Netw Learn Syst. 2023;34(12):9587–603. doi:10.1109/TNNLS.2022.3160699.
112. Bell J, Gascón A, Lepoint T, Li B, Meiklejohn S, Raykova M, et al. ACORN: input validation for secure aggregation. In: 32nd USENIX Security Symposium; 2023 Aug 8–11; Anaheim, CA, USA.
113. Keshavarzkalhori G, Pérez-Solà C, Navarro-Arribas G, Herrera-Joancomartí J, Yajam H. Federify: a verifiable federated learning scheme based on zkSNARKs and blockchain. IEEE Access. 2024;12:3240–55. doi:10.1109/ACCESS.2023.3347039.
114. Song DX, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In: Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000; 2000 May 14–17; Berkeley, CA, USA. p. 44–55. doi:10.1109/SECPRI.2000.848445.
115. Curtmola R, Garay J, Kamara S, Ostrovsky R. Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the 13th ACM Conference on Computer and Communications Security; 2006 Oct 30–Nov 3; Alexandria, VA, USA. p. 79–88. doi:10.1145/1180405.1180417.
116. Wu J, Zhang K, Wei L, Gong J, Ning J. Practical searchable symmetric encryption for arbitrary Boolean query-join in cloud storage. IEEE Trans Inf Forensics Secur. 2024;19:10086–98. doi:10.1109/TIFS.2024.3486002.
117. Xu P, Susilo W, Wang W, Chen T, Wu Q, Liang K, et al. ROSE: robust searchable encryption with forward and backward security. IEEE Trans Inf Forensics Secur. 2022;17:1115–30. doi:10.1109/TIFS.2022.3155977.
118. Xu L, Zheng L, Xu C, Yuan X, Wang C. Leakage-abuse attacks against forward and backward private searchable symmetric encryption. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security; 2023 Nov 26–30; Copenhagen, Denmark. p. 3003–17. doi:10.1145/3576915.3623085.
119. Zhang L, Wang J, Wu J, Wang Y, Sun SF. Violin: powerful volumetric injection attack against searchable encryption with optimal injection size. IEEE Trans Dependable Secure Comput. 2025;22(4):4103–15. doi:10.1109/TDSC.2025.3543248.
120. Vo V, Yuan X, Sun SF, Liu JK, Nepal S, Wang C. ShieldDB: an encrypted document database with padding countermeasures. IEEE Trans Knowl Data Eng. 2023;35(4):4236–52. doi:10.1109/TKDE.2021.3126607.
121. Stefanov E, van Dijk M, Shi E, Fletcher C, Ren L, Yu X, et al. Path ORAM: an extremely simple oblivious RAM protocol. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security—CCS'13; 2013 Nov 4–8; Berlin, Germany. p. 299–310. doi:10.1145/2508859.2516660.
122. Mathialagan S, Vafa N. MacORAMa: optimal oblivious RAM with integrity. In: Advances in cryptology—CRYPTO 2023. Cham, Switzerland: Springer Nature; 2023. p. 95–127. doi:10.1007/978-3-031-38551-3_4.
123. Vadapalli A, Henry R, Goldberg I. Duoram: a bandwidth-efficient distributed ORAM for 2- and 3-party computation. In: 32nd USENIX Security Symposium (USENIX Security 23); 2023 Aug 9–11; Anaheim, CA, USA. p. 3907–24.
124. Popa RA, Redfield CMS, Zeldovich N, Balakrishnan H. CryptDB: protecting confidentiality with encrypted query processing. In: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles; 2011 Oct 23–26; Cascais, Portugal. p. 85–100. doi:10.1145/2043556.2043566.
125. Zong C. The mathematical foundation of post-quantum cryptography. Research. 2025;8:0801. doi:10.34133/research.0801.
126. National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard: FIPS 203. Gaithersburg, MD, USA: NIST; 2024. doi:10.6028/NIST.FIPS.203.

127. Bos J, Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, Schanck JM, et al. CRYSTALS—kyber: a CCA-secure module-lattice-based KEM. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P); 2018 April 24–26; London, UK. p. 353–67. doi:10.1109/EuroSP.2018.00032.
128. National Institute of Standards, Technology. Module-lattice-based digital signature standard: FIPS 204. Gaithersburg, MD, USA: NIST; 2024.
129. Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, Schwabe P, Seiler G, et al. CRYSTALS-dilithium: a lattice-based digital signature scheme. IACR Trans Cryptogr Hardw Embed Syst. 2018:238–68. doi:10.46586/tches.v2018.i1.238-268.
130. National Institute of Standards and Technology. Stateless hash-based digital signature standard: FIPS 205. Gaithersburg, MD, USA: NIST; 2024. doi:10.6028/NIST.FIPS.205.
131. Bernstein DJ, Hülsing A, Kölbl S, Niederhagen R, Rijneveld J, Schwabe P. The SPHINCS + signature framework. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security; 2019 Nov 11–15; London, UK. p. 2129–46. doi:10.1145/3319535.3363229.
132. Alagic G, Bros M, Ciadoux P, Cooper D, Dang Q, Dang T, et al. Status report on the fourth round of the NIST post-quantum cryptography standardization process: NIST IR 8545. Gaithersburg, MD, USA: National Institute of Standards and Technology; 2025.
133. Moody D, Perlner R, Regenscheid A, Robinson A, Cooper D. Transition to post-quantum cryptography standards: NIST IR 8547 (Initial Public Draft). Gaithersburg, MD, USA: National Institute of Standards and Technology (NIST); 2024.
134. Cremonesi F, Vesin M, Cansiz S, Bouillard Y, Balelli I, Innocenti L, et al. Fed-BioMed: open, transparent and trusted federated learning for real-world healthcare applications. In: Federated learning systems: towards privacy-preserving distributed AI. Cham, Switzerland: Springer Nature; 2025. p. 19–41. doi:10.1007/978-3-031-78841-3_2.
135. Abbas SR, Abbas Z, Zahir A, Lee SW. Federated learning in smart healthcare: a comprehensive review on privacy, security, and predictive analytics with IoT integration. Healthcare. 2024;12(24):2587. doi:10.3390/healthcare12242587.
136. Al Badawi A, Alexandru A, Polyakov Y, Vaikuntanathan V. SoK: private LLM inference using approximate homomorphic encryption (Paper 2026/935) [Internet]. IACR Cryptol ePrint Arch; 2026 [cited 2026 Jan 1]. Available from: https://eprint.iacr.org/2026/935.