SoK: Private Transformer Inference Across Systems, Models, and Cryptography

Andes Y. L. Kei
Department of Information Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong

Sherman S. M. Chow
Department of Information Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong

Abstract

Private inference can protect user queries and model weights without trusted hardware or statistical privacy relaxations, but transformers combine large secret matrix multiplications, costly nonlinearities, and sequential autoregressive execution. Prior surveys organize the literature mainly by cryptographic backend, deployment setting, or supported operation, obscuring when techniques remain applicable or composable across execution phases, model adaptations, or security boundaries.

We systematize 58 cryptographic private transformer frameworks (2022–2026) across three interacting levels: systems (execution and optimization), models (cryptography–machine learning co-design and adaptation), and cryptography (secure realization of transformer operations). This analysis reveals two recurring cross-backend applicability constraints: optimizations do not transfer unchanged across execution phases when they require values not yet available, while data-dependent pruning, cache management, routing, and sparsity require private execution structure to be hidden, constrained, predicted, or disclosed. Against an output-only reference baseline, we identify four classes of security concerns affecting 5 frameworks and synthesize composition boundaries in maliciously secure designs. We find that 16 frameworks rely on empirically calibrated or distribution-specific mechanisms and 21 require additional training, limiting generalization and cross-framework comparability. We distill these findings into 12 open problems and 16 outlooks spanning protocol efficiency, cross-level co-design, dynamic execution, security, evaluation, and scalability.

1 Introduction

Transformers, the backbone of large language models (LLMs), handle sensitive queries and proprietary weights, making private inference a deployment concern. Homomorphic encryption (HE), secure multiparty computation (MPC), and function secret sharing (FSS) provide provable confidentiality without trusted hardware or statistical relaxations.

Yet, transformers are harder to evaluate cryptographically than earlier convolutional neural networks (CNNs). Multi-head attention requires large matrix multiplications over *secret* query, key, and value projections, while CNNs can often leverage more efficient plaintext-ciphertext operations when weights are public to the model owner. Core transformer functions, such as softmax and Gaussian error linear unit (GELU), involve expensive nonlinear operations that are poorly suited to HE and MPC. During inference, autoregressive decoding adds sequential, data-dependent dependencies and a growing key-value (KV) cache, limiting opportunities for parallel evaluation and increasing cryptographic overhead. As these challenges arise across different layers and execution stages, seemingly similar techniques at the cryptographic backend level can face different transformer-specific constraints.

Scope and Corpus Boundary. We focus on private computation for transformers using *purely cryptographic techniques*, excluding approaches relying on trusted execution environments (TEEs) [1], plaintext inference over permuted weights [2] or hashed inputs [3], and verifiable inference [4, 5].[1] We analyze design choices at three interacting levels: *systems* (execution phases and private execution structure), *models* (architectural and training adaptation to cryptographic costs), and *cryptography* (secure realization of transformer operations). Across these levels, we compare techniques along four dimensions relevant to their applicability and composition: target operations, phase availability of values, private structure exploited or disclosed, and composition across representations and security guarantees.

We apply a uniform eligibility rule within a frozen publication universe, counting only technically specified frameworks with integrated, materially distinguishable private-transformer designs. Eligibility defines corpus membership; analysis then classifies and compares techniques. Isolated components may be discussed but are not counted as separate entries; Appendix B details the full *CRAFTS* rule and audit procedure.

[1] We likewise exclude federated learning [6], differential privacy [7, 8], machine unlearning [9], membership inference [10], and model extraction attacks [11], which are not primarily cryptographic.

1.1 Contributions and Findings

Applicability and Composition. Prior surveys primarily classify private transformer frameworks by cryptographic backend or supported operation. We instead analyze when techniques remain applicable or composable across systems, models, and cryptography, revealing two cross-level constraints not systematically captured by prior classifications. First, *phase availability* constrains the transfer of optimizations across execution phases. CipherGPT [12] amortizes model-dependent preprocessing across future decoding steps because it is activation-independent, whereas NEXUS's [13] batched online optimization requires current activation matrices and therefore does not transfer unchanged to autoregressive decoding (§3.2). The same boundary appears in execution: attention-score pruning requires a computed utility signal, whereas speculative decoding supplies candidate future tokens early enough for parallel target evaluation (§6). Second, data-dependent techniques such as token pruning, KV-cache management, mixture-of-experts routing, and sparse execution share a *privacy-conditioned adaptivity* constraint: their efficiency depends on private structure, so existing designs hide decisions, constrain execution shape, predict structure, or disclose coarse information (§6–7). These constraints cut across backend-based classifications.

Evaluation Validity and Coverage Analysis. We analyze the assumptions underlying reported efficiency and accuracy results. Across the corpus, 16 systems rely on distribution-specific or empirically tuned approximations and 21 require extra training, while model and dataset coverage remain fragmented (§8.1). These dependencies limit generalization and meaningful cross-framework comparison.

Claim-Relative Security and Composition Analysis. We audit mechanisms relative to an output-only reference functionality. We observe four classes of potential security concerns affecting 5 frameworks and synthesize recurring composition boundaries in maliciously secure designs (§8.2–8.3).

We consolidate the resulting gaps into 12 open problems and 16 outlooks;[2] the agenda is summarized in Appendix A.

1.2 Related Work

The closest work is the concurrent[3] SoK of Chen *et al.* [14]. Their study is backend- and deployment-oriented, with added reproducibility analysis and controlled re-evaluation. The corpora overlap substantially but differ under our inclusion criteria: 46 of our 58 frameworks also appear in Chen *et al.*, 12 appear only in ours, while 15 works counted by Chen *et al.* do not enter our corpus; Appendix B gives the complete reconciliation. The two SoKs ask different questions. Chen *et al.* study how backend and deployment choices affect performance, reproducibility, and framework selection; we ask when techniques remain applicable or composable: whether their required values are available at the relevant execution phase, whether their decisions depend on private data, whether security guarantees survive representation or protocol changes, and whether their reported benefits depend on model adaptation or evaluation setting. These questions cut across backend boundaries. Their controlled re-evaluation is complementary to our analysis of applicability and composition.

Ng and Chow [15] systematize >50 privacy-preserving ML frameworks by cryptographic backends, primarily for CNNs. Li *et al.* [16] classify 16 works (up to early 2025) by supported operations, offering a narrower operation-centric perspective. Zbudila *et al.* [17] focus on MPC protocols and secret-sharing conversion, excluding pure-HE approaches and broader architectural or execution-level techniques. Andreoletti *et al.* [18] survey private LLM inference from a deployment perspective across cryptographic and non-cryptographic approaches, without systematic layer-level protocol or mechanism-level security analysis. Table 1 records each survey's perspective and analytical coverage; our distinction lies in analyzing when techniques remain applicable and composable across systems, models, and cryptography, rather than corpus breadth.

2 Preliminary

2.1 Transformer Cost Structure

A transformer maps *input tokens* or *image patches* to embedding vectors and processes them through stacked blocks containing *multi-head attention (MHA)*, *feed-forward networks (FFNs)*, and *layer normalization (LayerNorm)*. We distinguish *discriminative* transformers, such as BERT [19] and ViT [20], which process a fixed input in a single forward pass, from *generative* transformers, notably GPT [21], which generate tokens autoregressively with each step conditioned on prior outputs. For private inference, three costs organize the design space. First, linear projections multiply secret activations by model weights, while attention additionally multiplies secret query, key, and value matrices. Second, *softmax*, *GELU*, and LayerNorm require expensive nonlinear operations such as exponentiation, reciprocal, comparison, and *inverse square root*. Third, generative models add sequential *autoregressive execution* and a growing *key-value (KV) cache*, constraining parallelism and static cryptographic layouts. §3–6 follow these costs; Appendix C gives the architecture diagram, component equations, and representative model dimensions.

2.2 Deployment and Security Models

Privacy Goals and Leakage Convention. We distinguish three baseline goals: *query privacy*, which hides the client's query from other parties; *model privacy*, which hides the

[2] We use "Open Problem" for a concrete unresolved technical barrier and "Outlook" for a broader opportunity suggested by the systematization.

[3] Chen *et al.* first appeared as IACR ePrint 2026/491 in March 2026, during preparation of the May 2026 version of this manuscript.

Table 1: Coverage and Primary Analytical Perspective of Related Surveys

Work	Year	Perspective	Scope					Technique				Analysis				
			Trans	HE	MPC	Hybrid	Non-Crypto	Prot	Arch	Exec	Emerg	OP	App./Comp.	Sec Audit	Eval Gap	Re-Eval
Ng and Chow [15]	23	Backend	✗	✓	✓	✓	✗	■	✗	✗	✗	✓	✗	✗	✗	■
Li *et al.* [16]	25	Operational	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	■	✗
Zbudila *et al.* [17]	25	MPC-centric	■	✗	✓	✓	✗	✓	✗	✗	✗	■	✗	✗	✗	✗
Andreoletti *et al.* [18]	26	Deployment	✓	✓	✓	✓	✓	■	✗	■	✗	✗	✗	✗	✗	✗
Chen *et al.* [14]	26	Backend/deployment	✓	✓	✓	✓	✗	✓	■	✓	■	✓	■	✗	■	✓
This Work	26	Applicability/composition	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗

✓/■/✗ = Systematic/partial/not covered. Perspective = Primary analytical perspective or organizing axis ("backend" = crypto composition, *e.g.*, HE/MPC/hybrid). Scope: Trans = Transformer; HE = Homomorphic encryption; MPC = Multiparty computation; Non-Crypto = *e.g.*, TEE. Technique: Prot = Protocol-level; Arch = Crypto–ML co-design; Exec = Execution-level; Emerg = Emerging-paradigm coverage (cross-level). Analysis: OP = Open problem; App./Comp. = Applicability and composition analysis across levels; Sec Audit = Mechanism audit; Eval Gap = Validity/coverage gap; Re-Eval = New controlled evaluation (■ = limited). Markers describe treatment, not novelty/superiority; identical symbols are not directly comparable.

model owner's protected parameters; and *output privacy*, which reveals the inference result only to the client. These goals are relative to each framework's declared public information and protected state. Unless stated otherwise, our security audit uses an *output-only baseline*: beyond prescribed inputs, outputs, and public information, query- or model-dependent intermediate values are not revealed. Additional disclosure (*e.g.*, token-retention counts or aggregate sparsity) is therefore explicit leakage unless included in the ideal functionality or leakage profile; it is not automatically an attack.

Deployment Models. In *two-party computation (2PC)*, the client and server (or two outsourced servers) jointly evaluate the model. In *(2+1)PC*, an offline dealer generates preprocessing material and does not participate in online inference; security requires that it not collude with either online party. In *three-party computation (3PC)*, evaluation is distributed across three servers, typically with at most one corruption. In *oblivious inference*, the client and server participate directly; in *outsourced inference*, model and query are secret-shared to multiple evaluators. A protocol is *noninteractive* if the client sends an encrypted query and later receives an encrypted result without intermediate interaction.

Threat Models. Secure MPC is typically formalized by simulation relative to an ideal functionality: the adversary's real-protocol view must be simulatable from the prescribed inputs, outputs, public information, and explicitly modeled leakage. Most surveyed systems assume *semi-honest* adversaries, who follow the protocol but inspect the transcript; *malicious* adversaries may arbitrarily deviate.

2.3 Cryptographic Primitives and Backends

We use additive *secret sharing (SS)*, *replicated secret sharing (RSS)*, *homomorphic encryption (HE)*, *oblivious transfer (OT)*, *garbled circuits (GC)*, and *function secret sharing (FSS)* throughout. For notation, $[\![x]\!]$ denotes additive shares, $\langle x \rangle$ replicated shares, and $[x]$ an HE ciphertext. The surveyed cryptographic backends combine these primitives in five forms: *hybrid HE-MPC*, *OT-based MPC*, *Dealer-aided MPC*, *pure SS/FSS*, and *pure HE*. Their tradeoffs shape the comparisons below: hybrid systems pay representation-conversion overhead for flexibility; OT-based MPC systems offer low computation but high communication; dealer-aided and pure SS systems obtain fast online execution under additional non-collusion assumptions; pure HE provides noninteraction at substantially higher homomorphic cost. Appendix D reviews the underlying primitives and representative backend settings; Table 6 gives the full framework classification.

3 Linear Layer Protocols

Unlike CNNs, which are dominated by plaintext-ciphertext (pt-ct) matrix-vector products with plaintext weights and ciphertext activations, transformers involve both *pt-ct* and *ct-ct matrix-matrix multiplications*. MPC-based protocols (§3.1) offer low computation but high communication costs, while HE-based approaches (§3.2) invert this tradeoff.

3.1 Linear Operations on Secret Shares

Additive SS and RSS support noninteractive addition and public scalar multiplication at near-plaintext cost, but multiplying two secret-shared values (computing $[\![ab]\!]$ from $[\![a]\!]$ and $[\![b]\!]$ in additive SS, or $\langle ab \rangle$ from $\langle a \rangle$ and $\langle b \rangle$ in RSS) requires interaction. Additive SS uses *Beaver's multiplication triple* $([\![u]\!], [\![v]\!], [\![uv]\!])$ [22], generated via HE [23], OT [24], or a trusted dealer [25]. RSS instead uses *resharing* [26].

(R)SS (and BFV, §3.2) operates on exact integer arithmetic, yet transformer weights and activations are real-valued. The standard remedy is *fixed-point encoding*: a value $x_R \in \mathbb{R}$ is mapped to an ℓ-bit integer $x = \lfloor s \cdot x_R \rfloor \in \mathbb{Z}_{2^\ell}$ with scaling

factor $s = 2^f$ for a chosen precision f. Multiplying two f-precision values yields precision $2f$, so each product is then followed by a *truncation* step that restores precision to f.

3.2 Linear Operations on Ciphertexts

Throughout this section, $[\mathbf{XW}]$ denotes pt-ct matrix multiplication between the input $[\mathbf{X}]$ and the weight $\mathbf{W}$ in a linear layer, while $[\mathbf{AB}]$ denotes ct-ct multiplication in an attention layer. We focus on encoding and packing as the primary abstraction, treating higher-level protocols as consequences.

Coefficient Encoding. Coefficient encoding packs matrix entries into polynomial coefficients, enabling polynomial multiplication to evaluate multiple inner products without the costly rotations required by SIMD encoding.

Pt-Ct Multiplication. Cheetah [27] introduced this technique for private CNNs, and Iron [28] extended it to transformer pt-ct matrix multiplication with fewer ciphertexts. However, only a fraction of output coefficients contain useful results, wasting ciphertext capacity and the associated communication.

BumbleBee [29] uses homomorphic automorphisms to *interleave* sparsely populated ciphertexts into fewer densely packed ciphertexts, improving utilization and reducing coefficient communication at the cost of additional computation.

Ct-Ct Multiplication. Attention layers require multiplying two encrypted matrices. Since both operands depend on the client's input, these products cannot be reduced to efficient pt-ct multiplication. Early hybrid frameworks [28, 29] therefore convert HE ciphertexts into additive shares and evaluate ct-ct matrix multiplications with MPC protocols (§3.1).

SIMD Encoding. Later work adopted SIMD packing to maximize slot utilization. Common layouts include column, row, and diagonal packing, where each column, row, or diagonal is stored contiguously in a ciphertext [30].

Packing Optimized for Attention Structure. BOLT [31] introduces *HE-based ct-ct multiplication* specialized to the two packing layouts arising in attention, while using compact pt-ct packing and BSGS [32] to reduce rotations. THOR [33] instead uses a unified *diagonal-major* encoding across pt-ct and ct-ct operations, eliminating inter-layer repacking and supporting efficient transposition and SIMD parallelism. Powerformer [34] exploits *CKKS complex-number packing* for pt-ct multiplication and optimized ct-ct layouts, reducing key switchings by 35%–91% over Jiang *et al.* [35]. MOAI [36] co-designs *compatible packing formats* with the transformer computation graph, eliminating inter-layer repacking and using interleaved batching to remove internal ct-ct rotations. Appendix E gives the corresponding packing casework.

Weight-Structured Packing. Orthogonal work exploits structure in the weight matrices. PrivCirNet [37] enforces *block-circulant* weights, converting pt-ct matrix-vector multiplication into HE-friendly *1D cyclic convolutions*, while a latency-aware search selects layer-wise block sizes. This improves efficiency but introduces an accuracy-latency tradeoff that depends on the chosen block size as the block-circulant constraint is lossy relative to unconstrained weights.

HE-based linear protocols have evolved along two dimensions: *encoding design* (coefficient vs. SIMD packing) and *architecture awareness* (operation-level optimization vs. cross-layer co-design). Despite substantial empirical progress, a predictive cost model that jointly captures rotations, multiplicative depth, and slot utilization remains lacking.

Outlook 1. *Existing packing and encoding techniques are largely handcrafted and architecture-specific. Developing compiler-driven synthesis methods and formal lower bounds for rotations and repacking-aware layouts could shift packing design from ad hoc optimization to a systematic discipline.*

Outlook 2. *How can packing strategies scale efficiently with sequence length in long-context inference?*

Cross-Step Amortization. Several systems amortize repeated matrix multiplications against a shared weight matrix $\mathbf{W}$, but the phase in which amortization occurs determines whether it is compatible with autoregressive generation.

CipherGPT [12] targets autoregressive generation directly. For a predetermined budget of t response words and fixed $\mathbf{W}$, its preprocessing generates *subfield vector oblivious linear evaluation (sVOLE)* correlations [38] for t future multiplications. These correlations depend on $\mathbf{W}$ and t, but not on the activation matrices that will be produced during decoding. When the activation matrix $\mathbf{X}_j$ for the j-th token becomes available, the online phase simply consumes the corresponding precomputed correlation to evaluate the multiplication. Thus, sequential token dependence does not prevent CipherGPT from amortizing its preprocessing work across decoding steps. BumbleBee [29] nevertheless observes that the VOLE-based approach reaches low amortized communication only at a very large effective matrix dimension ($k \approx 10^7$ in its notation).

NEXUS [13] uses a different offline-online design under HE. Like CipherGPT, it preprocesses weight-dependent material for t multiplications without knowing the inputs. The key difference is the *dependency of the amortized optimization*. While CipherGPT amortizes activation-independent preprocessing, NEXUS's communication-compression optimization jointly processes the activation matrices of multiple multiplications. Its online phase concatenates $\mathbf{A}_0, \ldots, \mathbf{A}_{t-1}$ into $\mathbf{A}'$ and sends $\mathbf{A}' - \mathbf{R}$ in a single message. This particular t-way optimization therefore requires all t activation matrices when the online message is formed. It applies directly when the inputs are jointly available, but not unchanged to autoregressive steps whose later activations depend on earlier outputs.

The contrast suggests a *phase-availability* constraint: model-dependent work can span future steps if it is independent of their activations, while online batching cannot consume activations before preceding outputs produce them.

Open Problem 1. *What is the maximal cross-step amortization achievable in private autoregressive decoding? In*

particular, which model-dependent preprocessing and protocol state can be shared across future steps, and how much online work must remain per step?

Client-Side Outsourcing. Nimbus [39] shifts linear-layer HE computation to the client through row-wise encrypted weights and a *client-side outer-product* design, avoiding rotation-heavy server-side inner products and yielding 2.9–12.5× linear-layer speedups over prior work [29]. The tradeoff is client work proportional to the weight-matrix dimension, limiting this design when the client is substantially less capable than the server. Appendix E gives additional protocol detail.

Lightweight Client-Side Design. Although noninteractive protocols eliminate intermediate client interaction, systems such as NEXUS [13] still require client to encrypt and upload the full input matrix, incurring substantial cost.

Euston [40] reduces this overhead by decomposing the masking matrix via *singular value decomposition (SVD)*: $\mathbf{R} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$. The client encrypts only the diagonal of $\mathbf{D}$ and sends $\mathbf{U}$ and $\mathbf{V}$ in plaintext. The server reconstructs $[\mathbf{R}]$, precomputes $[\mathbf{RW}]$, and later uses it to unmask the client's ciphertext during inference. This significantly reduces client costs, yielding a 3100× client preprocessing reduction over NEXUS.

The efficiency gain [40] assumes that transmitting $\mathbf{U}$ and $\mathbf{V}$ in plaintext hides the masked input. A public preprint [41] argues that the SVD-based representation leaks the mask subspace, potentially allowing the server to recover private inputs from $\mathbf{U}$ and $\mathbf{V}$. As the attack has not been independently validated, its practical impact remains unclear.

Linear Layer Fusion. Beyond optimizing individual multiplications, recent work explores fusing multiple linear operators across layer boundaries. In hybrid HE-MPC frameworks, communication is dominated by conversions between HE and MPC domains at linear-to-nonlinear boundaries. Numerical approximation of nonlinear functions (see §4) often decomposes nonlinear layers into sequences of linear and nonlinear operators, creating opportunities where consecutive linear operators become adjacent with no intervening nonlinearity. Such operators can be fused when the output representation of one matches the input representation of the next.

BLB [42] exploits this observation through *operator-level fusion*. It fuses consecutive linear operators spanning nominal layer boundaries (*e.g.*, across the normalization-to-weight and weight-to-activation boundaries in FFN) and evaluates them as a single operator, eliminating intermediate HE-to-MPC conversions. BLB also differs from prior hybrid frameworks, which commonly adopt BFV-based HE. Instead, it uses CKKS to better tolerate noise growth during fused computation and introduces a secure CKKS–MPC conversion protocol. This design reduces communication by 4× compared with BOLT [31] and 2× compared with BumbleBee [29].

More broadly, BLB illustrates a cross-layer design constraint: nonlinear approximation determines which linear operators become adjacent, while the underlying cryptographic representation constrains whether the resulting fused computation can be implemented efficiently.

Outlook 3. *Can nonlinear approximations, cryptographic representations, and fusion plans be co-designed to maximize end-to-end benefit rather than optimized independently?*

3.3 Embedding Layer

One-Hot Input. An embedding layer selects row i of a weight matrix $\mathbf{W}_{\mathcal{T}} \in \mathbb{R}^{v \times d}$, where v is the vocabulary size and d is the embedding dimension. When the client encodes its token as a *one-hot vector* $\mathbf{e}_i \in \{0,1\}^v$, the lookup reduces to a matrix-vector product, and the secure matrix multiplication protocols discussed earlier apply directly. The one-hot representation, however, inflates the client's input from a single index to a v-dimensional vector, introducing communication overhead proportional to vocabulary size and mismatching the index-based interface of standard ML frameworks [43].

Index Input. Treating the embedding layer as an index-based lookup reduces the task to *private table lookup*: the server holds a private LUT $\mathcal{T}$ and the client holds a secret index i, both obtain a share of $\mathcal{T}[i]$ without revealing their inputs.

Three protocol families address this. The first is *direct indexing* on encrypted (or shared) LUT: the server sends an encrypted LUT $[\mathcal{T}]$ to the client, the client retrieves $[\mathcal{T}[i]]$ and sends masked $[\mathcal{T}[i]+r]$ back to the server. CipherGPT [12] and PrivTF [44] adopt this approach. The second keeps the table on the server and performs *oblivious row selection* via secure multiplexing (a protocol that evaluates a conditional selection without revealing the selection index to either party) [45]; PUMA [46] instantiates this with $O(v)$ equality tests, one per entry. The third, used by SHAFT [47], replaces equality tests with *dealer-generated correlated randomness*: a precomputed pair $(\llbracket r \rrbracket, \llbracket \mathbf{e}_r \rrbracket)$ of a random index and its secret-shared one-hot vector enables the parties to recover $\llbracket \mathbf{e}_i \rrbracket$ for the true index i in a single online round.

Among the surveyed frameworks, all three families require interaction: direct lookup transmits masked values (encrypted under the server's public key[4]), equality tests incur $O(v)$ communication, and correlated reconstruction needs one online round. Consequently, none of the surveyed noninteractive transformer inference frameworks supports index-based embedding (all ○ in "Emb" of Table 6).

Open Problem 2. *Can noninteractive private-transformer inference support index-based embedding with sublinear communication while remaining efficient for subsequent layers?*

[4] We note that this is unlike the noninteractive inference setting where all values are encrypted under client's public key.

4 Nonlinear Layer Protocols

4.1 Softmax

$\mathsf{Softmax}(\mathbf{x})[i] = e^{\mathbf{x}[i]}/\sum_j e^{\mathbf{x}[j]}$, the core nonlinearity in the attention layer, involves two nonlinear components, exponentiation e^x and reciprocal $1/x$, and requires maximum subtraction for numerical stability: $\mathsf{Softmax}(\mathbf{x}-\max(\mathbf{x})) = \mathsf{Softmax}(\mathbf{x})$.

Exponentiation is tractable as softmax inputs after max subtraction are non-positive; while unbounded below, highly negative inputs contribute negligibly. Approaches vary in approximation mechanisms and cryptographic backend compatibility.

Repeated squaring is the most broadly compatible family. *Limit characterization* approximates $e^x \approx (1+x/2^t)^{2^t}$ via t sequential squarings. Because it uses only additions and multiplications, it is naturally MPC-friendly and has been widely adopted across threat models [42, 46, 48]. *Scaling decomposition* [49], used by Euston [40], instead fits a polynomial to $e^{x/2^t}$ on a small domain and recovers e^x by t squarings of the result. The tighter argument range improves numerical stability, which is especially critical for pure-HE systems.

Table lookup [50] evaluates e^x from precomputed values indexed by input bit segments. Its low round complexity makes it the standard choice in FSS-based frameworks, where minimizing online latency is the primary goal [51, 52]. Conventional lookup costs scale linearly with table size [53], *i.e.*, exponentially in input bitwidth. Recent batch protocols achieve sublinear amortized cost for multiple queries to the same table [54, 55], yet existing systems process nonlinear activations independently and forgo these savings.

Outlook 4. *How can secure batch LUT protocols be systematically incorporated into private transformer frameworks to efficiently evaluate nonlinear functions?*

Empirically based approaches fit a low-degree polynomial over the empirically relevant input range. BOLT [31] decomposes the exponent into an integer part and a bounded residual based on an *empirically tuned* scaling factor, approximates the residual with a polynomial, and recovers the integer contribution via bit-shift. Nimbus [39] fits a *distribution-specific* polynomial tailored to the attention score distribution. Both reduce communication but rely on empirical assumptions that may limit transferability across models and tasks.

SecureGPT [56] computes limit characterizations under *multiplicative sharing*, making repeated squarings noninteractive. This approach is susceptible to precision loss and overflow in fixed-point arithmetic [57], and whether these issues can be resolved for transformers remains unclear.

Reciprocal is applied to the softmax denominator, a sum of exponentials whose range grows with sequence length, complicating fixed-domain approximation. Iterative methods nonetheless dominate in practice.

Goldschmidt's algorithm and *Newton's method* both converge quadratically: Goldschmidt permits parallel updates and fewer MPC rounds, while Newton's sequential, self-correcting updates better suppress finite-precision error [58]. Iron [28], BOLT [31], NEXUS [13], and THOR [33] adopt Goldschmidt, while PUMA [46] and East [59] adopt Newton's.

SOAL [60] adopts a *distribution-specific* approach, using outlier-aware training [61] to bound the denominator and enable polynomial initialization with two Newton's iterations. SecureGPT [56] avoids iteration by revealing the *masked denominator* rx in plaintext. Under fixed-point arithmetic, the public rx can remain secret-dependent, leaking beyond the output-only baseline unless this disclosure is modeled.

FSS systems use *LUTs*: SIGMA's table size scales with the input sequence length, Mosformer [52] uses *empirically tuned* rescaling for a fixed table size, and FuseFSS [62] generalizes with an *interval LUT* supporting interval-based queries.

Maximum. Secure comparison is the qualitative bottleneck of softmax: unlike exponentiation and reciprocal, which admit smooth-function approximation, comparison is non-polynomial and requires dedicated protocols.

Among interactive systems, the key axis is round complexity. *Tree reduction* [25, 28, 51] completes $O(m)$ comparisons in $O(\log m)$ rounds, where m is the input length in transformers; *sequential comparison* [29, 45, 59] takes $O(m)$ rounds.

Since HE-based comparison is substantially more expensive than OT-based alternatives [27, 63], pure-HE systems instead use *empirically determined* constant shifts [13, 33, 40, 48] or *omit normalization* [64–66] through architectural modification (§5), sacrificing generality for efficiency.

Open Problem 3. *Surveyed pure-HE designs that retain softmax typically replace exact maximum subtraction with empirical stabilization heuristics. Whether a softmax protocol can achieve noninteractive, efficient, and provably stable evaluation without distributional assumptions remains open.*

Computing Softmax as a Whole. Direct approximations avoid composing separate softmax primitives.

THE-X [67] uses an efficient learned approximation but incurs notable accuracy loss. SHAFT [47] instead uses an iterative *ordinary-differential-equation-based approximation* [68] that avoids private maximum computation. It enforces the convergence condition $\max(\mathbf{x}) - \min(\mathbf{x}) \le t$ through *empirically tuned* clipping bounds, where t is the iteration count.

Across component-wise and holistic designs, systems often rely on empirically tuned approximation or stabilization: distribution-specific polynomials [39], constant shifts [13], and fixed-range clipping [47]. These choices are fixed before inference rather than adapted to each private input.

Outlook 5. *Can empirically calibrated approximation or stabilization be adapted layer-wise or to private inputs at inference time without prohibitive overhead?*

Replacement. Crypto-friendly replacement removes exponentiation and comparison, but introduces accuracy degradation that must be recovered through training.

A canonical example is *2Quad*: $(\mathbf{x}[i]+c)^2/\sum_j(\mathbf{x}[j]+c)^2$, where c is an *empirically tuned* constant. MPCFormer [69]

pioneers 2Quad and applies *knowledge distillation (KD)*, fine-tuning the modified model against a softmax-based teacher's outputs to close the accuracy gap. RNA-ViT [70] generalizes to *higher polynomial degree* for improved accuracy at increased multiplicative depth, also relying on KD. SAL-ViT [71] and Powerformer [34] instead introduce *learnable parameters*, improving generalization without KD but still requiring modifications to the training pipeline.

Across all variants, polynomial parameters must be calibrated with access to the model and training data, precluding deployment on arbitrary pre-trained models.

4.2 GELU

$\mathsf{GELU}(x) = 0.5x(1 + \mathsf{Erf}(x/\sqrt{2}))$ is an activation in FFN. It behaves similarly to $\mathsf{ReLU}(x) = \max(x, 0)$ for large $|x|$, while exhibiting smoother curvature near 0. From a cryptographic perspective, the $\mathsf{Erf}(\cdot)$ term is the major obstacle.

Piecewise polynomial approximation is the dominant strategy in interactive systems. Reducing polynomial degree lowers multiplicative depth, but it introduces interval selection, requiring secure comparison. Early designs like PUMA [46] demonstrate this tradeoff by approximating GELU with a small number of low-degree polynomials over carefully chosen intervals, achieving high accuracy with moderate comparison overhead. This design pattern has since propagated across both hybrid (*e.g.*, BumbleBee [29]) and noninteractive frameworks (*e.g.*, NEXUS [13] and EncryptedLLM [48]).

Later designs reduce cost by exploiting even symmetry of $x\mathsf{Erf}(x/\sqrt{2})$ [31], *distribution-specific* approximation to high-probability ranges [39], or curvature-based breakpoints [72]; spline methods use many short intervals for accuracy but incur greater comparison overhead [12, 59].

Piecewise polynomial evaluation thus exposes a common tradeoff: accuracy comes from either higher-degree arithmetic or more intervals, with the latter requiring extra comparisons.

Composite polynomial approximation removes the need for secure interval selection by representing GELU or $\mathsf{Erf}(x)$ as a composition of polynomials. This structure is well suited to pure HE settings, where comparisons are expensive but arithmetic depth can be carefully managed.

Powerformer [34] uses a shallow, moderate-degree composition to limit multiplicative depth, while THOR [33] uses higher-degree compositions for greater accuracy at increased depth. Unlike piecewise methods, these designs improve accuracy through deeper arithmetic rather than interval refinement.

Fourier-series (FS) approximation expresses GELU-related functions as trigonometric expansions and suits (2+1)PC with preprocessed secure sine evaluation.

SecFormer [73] approximates $\mathsf{Erf}(x)$ over a bounded range, while SHAFT [47] applies FS only to a reformulated residual, reducing auxiliary arithmetic. Both achieve high accuracy with few terms, but their practicality hinges on efficient secure sine evaluation. The surveyed FS-based transformer instantiations use preprocessed sine in (2+1)PC; whether this approach remains efficient in dealer-free 2PC is unclear.

Open Problem 4. *Can FS-based nonlinear approximations be made practical in dealer-free 2PC despite the cost of secure sine evaluation?*

Tabular evaluation discretizes GELU into a small LUT indexed by the input value. This approach is particularly effective in FSS-based frameworks, where table lookup incurs low communication and avoids arithmetic depth.

SIGMA [51] exploits the even symmetry of $\mathsf{ReLU}(x) - \mathsf{GELU}(x)$ to approximate this residual using a compact table, reconstructing GELU by subtracting the lookup result from $\mathsf{ReLU}(x)$. Mosformer [52] adopts the same strategy.

Reformulation (Primitive Reduction). Several works replace the $\mathsf{Erf}(\cdot)$-based GELU formulation with forms better aligned with existing MPC primitives. Iron [28] and MOAI [36] adopt a *tanh-based* form $0.5x(1 + \tanh(\sqrt{2/\pi}(x + 0.044715x^3)))$, while SecureGPT [56], PrivTF [44], and Euston [40] instead employ a *sigmoid-based* form $x\mathsf{Sigmoid}(1.7017449x)$, where $\mathsf{Sigmoid}(x) = 1/(1 + e^{-x})$. These reformulations decouple GELU from $\mathsf{Erf}(x)$ and align it with better-studied secure protocols for tanh and e^x, at the cost of approximation error or reduced generality.

Open Problem 5. *Can alternative GELU formulations preserve model accuracy while reducing reliance on expensive nonlinear primitives in MPC?*

Replacement. Some systems replace GELU with cheaper functions, trading fidelity for efficiency. Early works (*e.g.*, [67, 74]) often substitute GELU with $\mathsf{ReLU}(x)$, though ReLU still requires relatively costly secure comparisons. Thus, more recent systems adopt low-degree polynomials, such as the quadratic [69, 75], to avoid branching. Both substantially reduce cost but incur non-negligible accuracy loss even after retraining because they alter the model's nonlinear behavior.

4.3 Inverse Square Root

LayerNorm requires secure evaluation of $1/\sqrt{x}$. Unlike $1/x$, iterative methods for $1/\sqrt{x}$ typically depend on a good initial estimate, while the lack of tight input bounds reduces the efficiency of LUT-based approaches.

Iterative Methods. Goldschmidt's [33, 42, 73] and Newton's methods [48, 59, 60] apply with the same tradeoff as for reciprocal, but require an additional squaring per iteration. Initialization strategy is the primary design axis, reflecting a tradeoff between protocol generality and deployment assumptions.

LUT-based initialization is general but incurs lookup overhead [50]; *exponentiation-based* initialization avoids LUTs but requires secure exponentiation [25]. SecFormer [73] narrows the input range using an *empirically tuned* scale and a constant initializer for Goldschmidt's algorithm to reduce initialization cost. THOR [33] applies *adaptive successive*

over-relaxation [76] within Goldschmidt's method to reduce iterations. SOAL [60] *suppresses outliers* during training, enabling a degree-3 polynomial initializer with only two Newton iterations. This comes at the cost of a non-standard training pipeline that cannot be deployed off-the-shelf.

Thus, convergence speed and initialization cost are coupled: designs with accurate initialization converge faster but impose protocol overhead, while those that avoid initialization shift the burden to input-range control or modified training phase.

Open Problem 6. *Is there a secure-computation-efficient, distribution-agnostic initialization for iterative* $1/\sqrt{x}$ *that provides a provable convergence guarantee?*

LUT-based approximation remains the standard in FSS-based systems, but the lack of useful input bounds for $1/\sqrt{x}$ drives up table size. SIGMA [51] uses a size-2^{13} interval-LUT, while Mosformer [52] reduces the domain via rescaling. Yet, its LUT size remains exponential in fixed-point precision f. Hybrid designs combining LUT initialization with iterative or polynomial correction could reduce LUT size without sacrificing accuracy, but remain unexplored in FSS-based systems.

Outlook 6. *Can hybrid designs combining LUT initialization with iterative or polynomial refinement reduce LUT size for* $1/\sqrt{x}$ *in FSS-based frameworks while preserving precision and incurring minimal overhead in online rounds?*

4.4 Exact Computation

Exact nonlinear evaluation is either expensive within cryptographic protocols or insecure when shifted to plaintext. GC is exact but scales poorly to transformers [77]; FSS efficiently supports comparison-based functions (*e.g.*, ReLU), while smooth nonlinearities (*e.g.*, GELU) are typically discretized or approximated. Selective revelation shifts nonlinear evaluation to plaintext: THE-X [67] decrypts between transformer blocks and evaluates in the clear, exposing exploitable intermediate representations [78]; Later systems mitigate leakage via shuffling-based defenses: PermLLM [79] employs oblivious permutation, and Fission [80] further partitions activations across parties. However, these defenses do not establish output-only simulation-based security: shuffling is vulnerable to model extraction [81], while Fission reports residual distribution leakage in its evaluated configuration [80].

5 Crypto–ML Co-Designed Architecture

MHA, LayerNorm, and FFN incur high costs under cryptographic protocols, motivating *cryptography–ML co-designed architectures* that adapt transformer building blocks to better align with cryptographic cost models.

5.1 Attention Variants and Reduction

Attention computes $\kappa(\mathbf{Q},\mathbf{K})\mathbf{V}$, where κ is a query-key similarity kernel and $\mathbf{V}$ is a value matrix. The standard attention uses $\kappa(\mathbf{Q},\mathbf{K}) = \mathsf{Softmax}(\mathbf{QK}^\top/\sqrt{d})$, where d is a model hyperparameter. MHA partitions the representation across H heads, applies separately projected attention in each head, and combines their outputs (Appendix C). Its cost stems from two axes. First, the kernel is non-polynomial: securely evaluating e^x, $1/x$, and $\max(\mathbf{x})$ is expensive (§4.1). Second, the number of nonlinear evaluations scales linearly with H and quadratically with sequence length m. Thus, existing approaches either simplify the kernel or reduce the operation count.

Softmax Elimination. This strategy removes non-polynomial operations, enabling evaluation via standard arithmetic circuits. MPCViT [82] and CryptPEFT [83] use *neural architecture search (NAS)* [84] to *selectively replace* the standard attention with *scaling attention* $(1/m)(\mathbf{QK}^\top)\mathbf{V}$ under a predefined cost budget. Similarly, PriViT [85] learns during training which softmax operations to retain and replaces the remainder with *square attention*, which squares $\mathbf{QK}^\top$ element-wise before multiplying by $\mathbf{V}$. PCFormer [86] *partitions tokens* during training, retaining softmax only for important tokens and substituting a *learned linear projection* for the rest. PolyTransformer [87] globally replaces softmax with a low-degree polynomial over attention scores. Despite reducing cryptographic overhead, these methods often incur task-dependent accuracy degradation, suggesting polynomial substitutes cannot fully preserve softmax-encoded information.

Nonlinear Kernel Replacement. These approaches retain nonlinear kernels but replace costly softmax components with cheaper alternatives. Privformer [88] and CipherFormer [89] replace softmax with element-wise *ReLU attention* [90], eliminating e^x, $1/x$, and $\max(\mathbf{x})$ for efficient evaluation via RSS or GC. ENSI [64] uses *sigmoid attention* [91] to retain a bounded nonlinearity while eliminating $\max(\mathbf{x})$, yielding higher accuracy than ReLU replacements. Rho et al. [65] isolate e^x via a *Gaussian kernel*, removing $1/x$ and $\max(\mathbf{x})$; this is competitive in pure HE settings but mandates retraining. SecViT [92] proposes *scale ReLU attention*, combining scaling and ReLU-based attention mechanisms to achieve a better efficiency-accuracy balance, albeit with modifications to the training pipeline. Overall, these methods balance accuracy and cryptographic cost: ENSI's sigmoid kernels avoid retraining, while Privformer's ReLU design assumes a pre-trained model with modified architecture; Gaussian kernels are more HE-friendly but generally require model adaptation.

Head Merging. Orthogonal to per-operation cost, the total nonlinear workload scales with H. Marill [93] clusters structurally similar heads and merges them into fewer heads with proportionally larger dimensions, reducing total nonlinear evaluations without altering the kernel.

Most work studies these strategies separately; broader co-designs that jointly reduce algebraic complexity and evalua-

tion frequency remain largely unexplored.

5.2 LayerNorm Simplifications and Variants

LayerNorm normalizes token activations using the token-wise mean and standard deviation, followed by a learned affine transform. Its main bottleneck is the computation of $1/\sqrt{x}$ during standard deviation calculation. Existing approaches address this challenge in three different ways.

Removal. MPCFormer [69] and SAL-ViT [71] *remove* LayerNorm to bypass the non-polynomial step, incurring non-negligible accuracy loss and mandating retraining.

Statistics Fixing. Replacing the input-dependent mean and standard deviation with constants collapses normalization into a "free" affine transform at inference. THE-X [67], Powerformer [34], and Rovida–Leporati [66] achieve this by substituting these statistics with *empirically tuned* or *learned* constants, while PolyTransformer [87] leverages *batch normalization* to fix training-accumulated running statistics at inference. Liu–Liu [74] keeps mean subtraction but substitutes the standard deviation with a *learned* constant, eliminating the non-polynomial $1/\sqrt{x}$. These methods typically require retraining and may generalize poorly across downstream tasks.

Fusion-Friendly Variant. Traditional transformers (*e.g.*, BERT [19]) use *post-norm* blocks, applying LayerNorm after MHA and FFN. PrivTF [44] instead adopts a *pre-norm* architecture, placing LayerNorm directly before MHA and FFN in transformer blocks. This variant enables fusion of LayerNorm's affine transform with subsequent matrix multiplications, reducing the number of secure multiplications.

5.3 Cryptography-Aware Activation Variants

The standard transformer architecture employs GELU as the activation function of FFN. While GELU mimics ReLU for large $|x|$, its high approximation cost near 0 motivates replacing it with more efficient ReLU-like functions.

ReLU. Several frameworks replace GELU with ReLU, including Primer [77], CipherFormer [89], and Privformer [88]. ReLU can be efficiently realized via secure comparison or GC, but may incur significant accuracy loss.

ReLU-Like Activations. RNA-ViT [70] and SecViT [92] adopt *leaky ReLU* $\mathsf{LReLU}(x) = \max(\alpha x, x)$ with a small *empirically tuned* $\alpha \in (0,1)$, balancing accuracy and latency. ENSI [64] and SOAL [60] use the *sigmoid linear unit* $\mathsf{SiLU}(x) = x\mathsf{Sigmoid}(x)$. Although this design improves accuracy, it incurs higher computational cost. Rho *et al.* [65] employ *gated linear unit* $\mathsf{GLU}(\mathbf{x}) = \mathbf{x}_0 * \mathsf{Sigmoid}(\mathbf{x}_1)$, where $*$ denotes element-wise product and $\mathbf{x} = \mathbf{x}_0 \| \mathbf{x}_1$. This approach retains the accuracy benefits of sigmoid-based activations while reducing the hidden dimensionality and overall cost.

Overall, most work treats activations in isolation, focusing on cryptography-aware variants for better accuracy-efficiency tradeoffs. Joint architecture-approximation co-design remains underexplored, limiting opportunities for efficiency gains.

Outlook 7. *Can activation design be leveraged to simplify other costly nonlinear components, such as softmax attention and normalization, by controlling the dynamic range or distribution of intermediate representations?*

Softmax stabilization typically requires either expensive secure $\max(\mathbf{x})$ or empirically tuned constant shifts (§4). One concrete direction is to co-design *upper-bounded* ReLU-like activations, such as $\mathsf{ReLU6}(x) = \min(\mathsf{ReLU}(x), 6)$, with surrounding layers to obtain tighter bounds on downstream softmax inputs, potentially simplifying stabilization.

5.4 Low-Precision Weights

Quantization compresses models by mapping real-valued weights and activations x_R to low-bit integers using a scale $s > 0$ and integer offset b, typically as $x = \text{round}(x_R/s) + b$. In *static* quantization, s and b are fixed before inference via calibration [94], avoiding input-dependent scaling. Ditto [75] uses the fixed-point special case of *dyadic* quantization ($s = 2^{-f}$) with layer-specific scales; scale conversion then reduces to shifts or truncation, substantially reducing MPC cost.

PPML-Quantized [95] and Lu *et al.* [96] remove the public-parameter constraint to support arbitrary scales in a fully private setting. In 2PC, PPML-Quantized evaluates all layers via *private LUTs*, directly applying quantized functions to bypass repeated encode-decode overhead at the cost of severe preprocessing communication. Lu *et al.* extend this to 3PC via RSS, which admits more efficient private LUT evaluation and reduces that communication cost. All surveyed systems target static quantization. *Dynamic* quantization instead derives quantization parameters from the input at inference time [94]. Under the output-only baseline of §2.2, these input-dependent parameters must therefore be computed privately.

Open Problem 7. *Can dynamic quantization be realized privately while retaining its efficiency gains despite the cost of deriving input-dependent quantization parameters?*

Ternary weights $\{-1, 0, 1\}$ remove the multiplicative cost of linear layers by reducing multiplications to conditional additions or subtractions. ENSI [64] exploits this via a transformer natively trained with ternary weights [97] while keeping full-precision activations. Consequently, pt-ct matrix multiplication reduces to a sequence of additions and subtractions, eliminating the dominant per-layer multiplicative cost.

The strategies from each subsection above remain largely siloed, and their compositional interactions (*e.g.*, pairing ternary weights with LayerNorm removal, or scaling attention with ReLU activations) lack systematic evaluation.

Outlook 8. *Not all co-design strategies are paradigm-neutral. For example, ReLU-based attention aligns naturally with MPC, where secure comparison is relatively efficient, while statistics fixing appears primarily in HE-based designs, where*

it reduces LayerNorm to affine computation. Systematically characterizing which combinations of modifications are best suited to HE versus MPC cost models would enable more principled and application-specific co-design.

6 Inference Execution Protocols

Execution-level protocols optimize end-to-end private transformer inference beyond layer evaluation and architecture.

6.1 Input Token Pruning

Attention layer computation scales quadratically with sequence length, severely amplifying cryptographic overhead per operation. Token pruning discards less informative tokens before or during evaluation to reduce downstream algebraic costs. The primary design axis is *pruning granularity*: whether the discard ratio is predetermined or adaptively computed from the input. BOLT [31] applies *one-time*, *fixed-ratio* pruning at the first attention layer, discarding tokens below the median attention score via oblivious ranking. Because the retained count is fixed in advance, this design does not disclose an input-dependent pruning ratio, but it rigidly discards the same fraction regardless of input complexity. CipherPrune [98] applies *adaptive layer-wise* pruning by privately comparing cumulative attention scores against a threshold and revealing only the retained token count N' to the evaluating parties. This adaptability improves efficiency for inputs heavily populated by uninformative tokens, but revealing N' constitutes additional structured, input-dependent leakage under the output-only baseline of §2.2. CipherPrune further reduces per-token evaluation cost by dynamically assigning lower-degree polynomial approximations to marginally informative tokens, trading accuracy for speed.

Outlook 9. *Existing pruning systems largely rely on attention scores computed during inference, requiring substantial secure computation before any tokens can be removed. Can we develop lightweight predictors that estimate token utility before expensive attention evaluation, while preserving accuracy and avoiding additional input-dependent disclosure?*

6.2 KV Cache Management

KV caching accelerates transformer inference by reusing previously computed keys and values, avoiding redundant attention computations. During autoregressive generation, the cache grows linearly with sequence length, increasing both memory and computational costs. To address this, cache eviction methods retain only the most relevant tokens, typically using approximate query-key similarity.

In cryptographic inference, KV caching poses additional challenges. Pure-HE frameworks [13, 33, 36] often rely on fixed cache layouts, making dynamic cache resizing difficult. Cachemir [99] overcomes this limitation with an *interleaved replicative encoding* scheme that supports incremental insertion of encrypted key-value pairs while preserving compatibility with fixed-layout HE operations.

MPC-based systems provide greater flexibility for cache management. MPCache [100] performs a *one-time* MPC-based eviction at the first generation step, removing cached tokens with low cumulative attention scores. It further reduces cost through *dynamic token selection*, using MPC-based hierarchical clustering and approximate dot-product similarity to retain only important cached tokens during generation.

Eviction discards selected KV entries, whereas compression retains past context more compactly, *e.g.*, via token merging or quantized key-value storage. Despite extensive study in plaintext inference [101], secure KV cache compression remains underexplored: compression requires data-dependent operations to identify redundant or low-importance cache entries, such as similarity comparisons, nearest-neighbor searches, or ranking. Under cryptographic protocols, these operations translate into private comparisons, argmin operations, or other conditional computations that can be substantially more expensive than the dot-product-based eviction or fixed-layout appends supported by current frameworks.

Outlook 10. *Can lossy KV cache compression hide its decisions without requiring full-cache private search or ranking?*

6.3 Next-Token Selection

During generative inference, the model selects the next token from output logits according to a decoding strategy. Greedy decoding selects the highest-logit token by default via secure argmax, requiring $O(v)$ comparisons over a vocabulary of size v but often produces low-diversity text.

CipherGPT [12] replaces greedy decoding with *top-K sampling* [102], which samples a token from the K highest-logit candidates. The central insight is that top-K sampling incurs no asymptotic overhead over greedy decoding: oblivious partial sorting [103] isolates the top-K partition in $O(v)$ expected comparisons, and sampling from K candidates requires only $O(K)$ additional comparisons. Since $K \ll v$ in practice, top-K sampling can improve output diversity at the same asymptotic cost as greedy argmax.

MERGE [104] takes an orthogonal approach, bypassing token selection entirely by feeding the hidden output state directly as the next input embedding, eliminating the $O(v)$ argmax at the cost of substantial accuracy degradation.

Outlook 11. *Existing approaches either adopt plaintext decoding strategies, such as greedy decoding and top-K sampling, or bypass token selection at substantial accuracy cost. Can decoding strategies be co-designed with cryptographic cost models to reduce token-selection cost while preserving output quality?*

6.4 Speculative Decoding

Speculative decoding [105] accelerates inference by proposing K candidate tokens via a lightweight draft model, allowing the target model to verify them in a single parallel pass. A stochastic acceptance criterion preserves the target's exact output distribution.

Post [106] adapts this to private inference: the client locally generates K secret-shared candidate tokens using a public draft model, and the server evaluates the private target model on them in parallel. An MPC protocol accepts the longest draft prefix matching the target distribution, reducing sequential target evaluations when multiple candidates are accepted. From a phase-availability perspective, the draft model makes candidate future tokens available early for parallel target evaluation, with correctness enforced by subsequent verification.

The public draft model is obtained via KD from the target and is intentionally disclosed, so it belongs to the declared public instance rather than constituting transcript leakage by itself. Because it is derived from the private target, however, the resulting relaxation of model privacy is not characterized, and the draft may also expose properties of the target's training data beyond those intended for disclosure.

Open Problem 8. *What model-privacy definition and leakage profile should govern public draft models distilled from private targets, and how should training-data exposure be modeled? Can draft-model construction bound these disclosures while retaining the benefits of speculative decoding?*

7 Protocols for Emerging ML Paradigms

7.1 Parameter-Efficient Fine-Tuning (PEFT)

PEFT adapts a pre-trained transformer by training a small *adapter* while leaving the *backbone* (containing the vast majority of parameters) fixed. The two dominant PEFT approaches are *low-rank adaptation* (LoRA), which adds a low-rank decomposition to selected weight matrices [107], and *adapter modules*, which insert lightweight trainable layers between transformer blocks [108]. This architecture naturally enables a cryptographic partition: if the backbone is public, only the adapter requires model-parameter protection.

Existing systems differ in the extent to which they exploit this partitioning and in whether adapter weights are treated as private. Under the convention of §2.2, an explicitly public backbone is part of the public instance; model privacy then applies only to the adapter or other parameters designated private. Marill [93] takes the most aggressive approach: the entire backbone is executed in the clear, while private computation is restricted to a single LoRA layer in the *final* transformer block. This approach maximizes efficiency but limits expressivity. Rho *et al.* [65] instead propose a *noninteractive* HE-based design that keeps the pretrained backbone in plaintext at the server while encrypting the user's LoRA weights across all adapted layers. Its efficiency stems from LoRA's low-rank structure, which reduces the cost of the dominant encrypted matrix multiplications.

CryptPEFT [83] co-designs adapter architecture with cryptographic protocol, rather than fitting a fixed adapter. Backbone and adapter computations are decoupled so that the backbone pass requires only *one-way* client-to-server communication; adapter insertion points are chosen via NAS to minimize the number of nonlinear layers in the inference circuit; and standard attention and GELU are replaced with linear attention and ReLU to reduce interactive depth. The key gains come from restructuring the adapter to cut cryptographic cost.

All three systems exploit PEFT to reduce private computation through different choices of adapter placement, protected parameters, and cryptographic execution. If adaptation itself must be private, PEFT limits trainable parameters and parameter updates to a small subset, whereas full fine-tuning additionally requires gradients and updates for the full model.

Outlook 12. *Can co-design across cryptographic protocols, model architectures, and training algorithms narrow the efficiency gap between private full fine-tuning and parameter-efficient adaptation at transformer scale?*

7.2 Mixture-of-Experts (MoE) Architecture

MoE layers [109, 110] replace a monolithic FFN with a bank of parallel *experts*, routing each token to a learned subset. A softmax gate computes routing scores to activate the top-K experts (typically K=2) and combines their outputs via the corresponding weights. Private inference introduces two challenges: data-dependent routing mandates oblivious expert selection, and the gating softmax is a transcendental function whose private evaluation requires garbled circuits or deep polynomial approximation. Existing systems address the first challenge by enforcing *balanced routing*, which constrains each expert to process exactly t tokens and reduces oblivious routing to a fixed-width permutation.

CryptoMoE [111] implements this via secure top-K selection and index-to-one-hot encoding to mask which experts are activated. SecMoE [112] further removes the softmax by assuming uniform expert weights, discarding learned confidence scores in exchange for purely linear gating. Crucially, both systems assume that all experts are available within a single trust domain. They do not consider a stronger setting in which experts are distributed across distinct trust domains [113].[5] Balanced routing fixes the per-expert load in the single-trust-domain setting, but does not by itself hide the destination expert once routing crosses trust domains.

Open Problem 9. *How can oblivious routing be realized across mutually distrusting servers without communication overhead proportional to the total number of experts?*

[5] This differs from outsourced inference (§2.2), where all servers hold shares of every expert; here, different experts are hosted by different servers.

7.3 Sparse LLMs

Transformer *activation sparsity* [114] often yields near-zero neuron outputs that contribute negligibly to final results.

Plaintext algorithms exploit sparsity by skipping near-zero activations, but revealing the activation positions would add input-dependent structural leakage under the output-only baseline. Comet [115] mitigates this tension by decoupling the sparsity pattern from its spatial positions. It trains an offline *lightweight predictor* to estimate near-zero neurons, performs sparse matrix multiplications based on this estimated distribution, and uses oblivious shuffling to hide exact neuron positions while revealing only the aggregate zero count. This controlled-disclosure design improves efficiency but depends on predictor accuracy and threshold calibration to prevent severe accuracy degradation. Comet sidesteps, rather than resolves, the underlying dilemma: directly skipping operations according to the true input-dependent sparsity pattern requires either exposing that pattern or hiding the decision obliviously.

Open Problem 10. *Can a protocol exploit input-dependent sparsity with sublinear cost while hiding its locations, without predictive approximation or additional structural leakage?*

Data-Dependent Structure. Token pruning (§6.1), KV-cache management (§6.2), MoE routing (§7.2), and sparse execution (§7.3) all exploit private, data-dependent structure. Existing designs realize these gains by evaluating decisions obliviously, constraining execution to fixed or bounded shapes, predicting useful structure, or disclosing coarse structural statistics. Read together, these mechanisms expose a privacy-conditioned adaptivity constraint: adaptivity, cryptographic cost, and leakage cannot be analyzed independently.

7.4 Multimodal LLMs

Private multimodal LLM inference primarily reuses cryptographic building blocks for vision and language transformers, together with modality-specific input processing and multimodal integration. The only framework in our corpus targeting this setting, PrivMLLM [116], combines protocol-level improvements with systems optimizations such as mixed-precision computation and modality-aware bitwidth assignment, without introducing new cryptographic primitives. Whether cross-modal fusion creates composite privacy risks absent in isolated modalities remains uncharacterized.

8 Practical Issues and Open Problems

8.1 Validity, Retraining, and Evaluation Gaps

Empirical approximations and retraining improve efficiency, but they introduce hidden costs that are often excluded from evaluation, limiting fair cross-framework comparison.

Generalization of Empirical Approximations. Many frameworks exploit empirical or distribution-specific properties, such as concentrated activation inputs. While these heuristics often show minimal accuracy loss on limited benchmarks,[6] their robustness across larger architectures, longer contexts, or diverse tasks, remains largely unverified.

Open Problem 11. *What positive/negative generalization guarantees hold for distribution-specific approximations in private inference? Under what conditions do they fail?*

Retraining and Data Availability. Retraining methods (*e.g.*, KD, NAS) improve efficiency and accuracy but require representative downstream data, which may be unavailable or confidential. When such data can be used but not disclosed, secure retraining offers a potential path forward. However, KD and NAS within MPC remain unexplored.

Outlook 13. *Can KD and NAS be realized efficiently within MPC for retraining without exposing downstream data?*

Benchmark Coverage and Comparability. Table 7 reveals fragmented evaluation despite widespread empirical and retraining techniques: among our corpus of 58 works, 16 use empirical approximations and 21 require retraining, yet only 25 evaluate on BERT-base (110M parameters), 8 on BERT-large (340M), 11 on GPT-family (GPT-2-base, and GPT-2-XL for Comet [115] only), 6 on billion-parameter models (LLaMA2-7B, LLaMA3-8B, GPT-2-XL). Results may not transfer across scales or tasks, *e.g.*, MPCFormer [69] degrades on BERT-large despite acceptable BERT-base performance [73]. Vision and multimodal tasks appear in only 10 works, and even within language benchmarks most evaluations cover only a subset of GLUE [117] tasks, making direct comparison unreliable when reported subsets differ. Vision evaluations primarily focus on CIFAR-10/100 [118] and Tiny-ImageNet [119]. Consequently, Table 6's Acc column should be interpreted only as fidelity relative to each work's own plaintext baseline rather than a cross-framework ranking. Identical bins across rows do not imply comparable performance.

Outlook 14. *What evaluation protocols and benchmark suites would enable fair cross-framework comparison, accounting for the hidden costs of empirical and retraining approaches?*

A standardized evaluation framework should make the comparison boundary explicit by controlling for the security and deployment setting while covering multiple *model scales* and *task categories*. It should report the *hardware/software* stack, *network* conditions, and an accounting boundary separating preprocessing from online inference. Evaluations should report both *performance metrics* (*e.g.*, end-to-end latency, communication, peak memory, accuracy) and *deployment costs* (*e.g.*, retraining effort and requirements).

8.2 Security Issues in Prior Systems

Relative to the output-only baseline in §2.2, some systems disclose unmodeled intermediate values or rely on heuristic masking and permutation defenses. Table 4 summarizes

[6] Appendix F for a summary of commonly evaluated models and datasets.

the identified concerns. We distinguish *established* from *preliminary* concerns. Established concerns are supported either *structurally*, when the mechanism or an explicit witness establishes leakage beyond the output-only reference functionality, or *empirically*, when experiments demonstrate the leakage or its security impact. Preliminary concerns are externally reported but remain independently unverified.

These cases yield three recurring lessons. First, randomness alone does not ensure privacy: nonuniform or correlated masks may retain exploitable information. Second, all disclosed values (*e.g.*, intermediate activations, shuffled data) must be explicitly modeled. Third, component-level cryptanalysis should be propagated cautiously: vulnerabilities identified in a specific component do not automatically extend to all frameworks. For example, Li *et al.* [120] identify a security flaw in probabilistic truncation protocols (*e.g.*, [26]), arising from the coupling of masking randomness with the probabilistic rounding error. Subsequent work has adopted alternative truncation functionalities and constructions that do not inherit the same issue. Thus, security claims must be assessed against both the concrete protocol and the ideal functionality it realizes. Accordingly, Table 6 marks preliminary issues with **?** and established issues with **!**; unmarked rows mean no concern identified by this audit, not security certification.

Outlook 15. *As §6–7 show, systems often disclose structural information (e.g., token-retention counts or sparsity metrics) to reduce the cost of exploiting data-dependent transformer execution. While such leakage can be incorporated into the ideal functionality, doing so weakens the output-only baseline and leaves its privacy cost unquantified. Developing quantitative leakage-accounting frameworks remains open.*

8.3 Malicious-Security Boundaries

Among the corpus, 4 systems establish malicious security in at least one declared instantiation: SHARK [121] provides maliciously secure (2+1)PC with preprocessing; Privformer [88] achieves malicious security in honest-majority 3PC under information-theoretic correlated randomness; Mosformer [52] uses verifiable DPFs (VDPFs) for maliciously secure 3PC; SMASH [122] provides malicious security for its n-party ($n \geq 2$) LUT pipeline, while its optional arithmetic-Boolean share conversion (A2B)/B2A variants are semi-honest.

These systems rely on backend-specific integrity mechanisms. SHARK uses FSS/interactive FSS, Mosformer employs VDPF-based lookup protocol, SMASH combines lightweight zero-knowledge proofs (ZKPs) with maliciously secure share conversion, and Privformer composes secure subprotocols with information-theoretic correlated randomness.

Three boundaries emerge from prior systems. First, malicious checks focus primarily on nonlinear operations: interactive FSS protocols in SHARK, VDPF-based comparison/lookup in Mosformer, and ZK-verified lookup in SMASH. Second, representation conversions must maintain the same security guarantee: Mosformer includes secure modulus conversion, and SMASH's arithmetic–LUT conversions are part of its malicious-security path, while SMASH's optional A2B/B2A variants are only semi-honest. Third, the setup and composition argument must match the deployed implementation: SHARK assumes correlated preprocessing, and Privformer's benchmarked PRNG instantiation is outside its stated concurrent-composition argument. Overall, maliciously secure transformer inference is feasible but remains fragmented across backends, trust models, and protocol compositions.

Outlook 16. *Can private-transformer inference obtain end-to-end malicious security while preserving the optimizations that make semi-honest systems efficient?*

Promising ingredients include authenticated or interactive FSS, VDPFs, lightweight ZKPs, and maliciously secure share-conversion techniques that compose cleanly across the entire inference pipeline under one declared threat model.

8.4 Scalability to Large Language Models

Scaling private inference to billion-parameter models makes memory capacity and bandwidth first-order constraints, particularly for HE. For CKKS, de Castro *et al.* [123] show low arithmetic intensity and memory-bandwidth bottlenecks on CPU/GPU platforms. Recent HE systems address large-model memory pressure: Cerium [124] uses compressed plaintext representations and host–GPU/multi-GPU orchestration to support HE inference for LLaMA3-8B [125]. SS-based frameworks face different memory pressure from large activations and nonlinear-approximation state. Cross-backend memory-bounded execution remains underexplored.

Open Problem 12. *Can private inference reduce peak working memory, for example through blockwise or streaming execution, without simply shifting the bottleneck to communication or recomputation at billion-parameter scale?*

9 Conclusion

Across 58 frameworks, two recurring constraints shape applicability and composition across system, model, and cryptography. Phase availability limits cross-phase optimization: designs either avoid dependencies on unavailable values or, as in speculative decoding, supply provisional values for later verification. Data-dependent pruning, cache management, routing, and sparsity derive efficiency from private execution structure; existing designs hide, constrain, predict, or explicitly disclose that structure. Our security and evaluation audits further expose limits from leakage, composition, approximation assumptions, retraining, and benchmark coverage. Together, these findings motivate cross-level co-design and ground the 12 open problems and 16 outlooks in Appendix A.

References

[1] M. Yuan, L. Zhang, Y. Cheng, M.-H. Song, G. Xing, and X. Li, "STIP: Three-party privacy-preserving and lossless inference for large transformers in production," in *NDSS*, 2026.

[2] J. Luo, G. Chen, Y. Zhang, S. Liu, H. Wang, Y. Yu, X. Zhou, Y. Qi, and Z. Xu, "CENTAUR: bridging the impossible trinity of privacy, efficiency, and performance in privacy-preserving transformer inference," in *ACL*, 2025, pp. 22 751–22 770.

[3] A. Mishra, M. Li, and S. S. Deo, "SentinelLMs: Encrypted input adaptation and fine-tuning of language models for private and secure inference," in *AAAI*, 2024, pp. 21 403–21 411.

[4] H. Sun, J. Li, and H. Zhang, "zkLLM: Zero knowledge proofs for large language models," in *CCS*, 2024, pp. 4405–4419.

[5] W. Qu, Y. Sun, X. Liu, T. Lu, Y. Guo, K. Chen, and J. Zhang, "zkGPT: An efficient non-interactive zero-knowledge proof framework for llm inference," in *USENIX Sec.*, 2025, pp. 2045–2063.

[6] Y. Ding, P. Mai, J. Zhang, S. S. M. Chow, M. Du, and Y. Pang, "Shared Spotlight Meridian: Distributed sparse pseudorandom functions for scalable federated learning," in *S&P*, 2026, pp. 2269–2287.

[7] M. Du, X. Yue, S. S. M. Chow, and H. Sun, "Sanitizing sentence embeddings (and labels) for local differential privacy," in *WWW*, 2023, pp. 2349–2359.

[8] M. Du, X. Yue, S. S. M. Chow, T. Wang, C. Huang, and H. Sun, "DP-Forward: Fine-tuning and inference on language models with differential privacy in forward pass," in *CCS*, 2023, pp. 2665–2679.

[9] J. Yao, E. Chien, M. Du, X. Niu, T. Wang, Z. Cheng, and X. Yue, "Machine unlearning of pre-trained large language models," in *ACL*, 2024, pp. 8403–8419.

[10] Y. He, B. Li, L. Liu, Z. Ba, W. Dong, Y. Li, Z. Qin, K. Ren, and C. Chen, "Towards label-only membership inference attack against pre-trained large language models," in *USENIX Sec.*, 2025, pp. 1609–1628.

[11] N. Carlini, F. Tramèr, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. B. Brown, D. Song, Ú. Erlingsson, A. Oprea, and C. Raffel, "Extracting training data from large language models," in *USENIX Sec.*, 2021, pp. 2633–2650.

[12] X. Hou, J. Liu, J. Li, Y. Li, J. Zhang, W. Lu, C. Hong, and K. Ren, "CipherGPT: Secure two-party GPT inference," *IEEE Transactions on Dependable and Secure Computing*, vol. 23, no. 3, pp. 6748–6762, 2026.

[13] J. Zhang, X. Yang, L. He, K. Chen, W. Lu, Y. Wang, X. Hou, J. Liu, K. Ren, and X. Yang, "Secure transformer inference made non-interactive," in *NDSS*, 2025.

[14] Y. Chen, T. Lu, Z. Tang, B. Zhang, Z. Shi, Y. Luan, and Z. Wang, "SoK: Private transformer-based model inference," in *USENIX Sec.*, 2026.

[15] L. K. L. Ng and S. S. M. Chow, "SoK: Cryptographic neural-network computation," in *S&P*, 2023, pp. 497–514, full version at https://sokcryptonn.github.io/.

[16] Y. Li, X. Zhou, Y. Wang, L. Qian, and J. Zhao, "Private transformer inference in MLaaS: A survey," arXiv 2505.10315, 2025.

[17] M. Zbudila, A. Suresh, H. Yalame, O. Mirzamohammadi, A. Abidin, and B. Preneel, "SoK: Connecting the dots in privacy-preserving ML - systematization of MPC protocols and conversions between secret sharing schemes," Cryptology ePrint 2025/1679, 2025, presented at TPMPC 2026.

[18] D. Andreoletti, A. Rudi, E. Carpanzano, F. Lelli, and T. Leidi, "Privacy-preserving LLM inference in practice: A comparative survey of techniques, trade-offs, and deployability," Cryptology ePrint 2026/105, 2026.

[19] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT (1)*, 2019, pp. 4171–4186.

[20] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *ICLR*, 2021.

[21] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018, OpenAI report.

[22] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *CRYPTO*, 1991, pp. 420–432.

[23] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference service for neural networks," in *USENIX Sec.*, 2020, pp. 2505–2522.

[24] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *S&P*, 2017, pp. 19–38.

[25] B. Knott, S. Venkataraman, A. Y. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten, "CrypTen: Secure multi-party computation meets machine learning," in *NeurIPS*, 2021, pp. 4961–4973.

[26] P. Mohassel and P. Rindal, "Aby^3: A mixed protocol framework for machine learning," in *CCS*, 2018, pp. 35–52.

[27] Z. Huang, W. Lu, C. Hong, and J. Ding, "Cheetah: Lean and fast secure two-party deep neural network inference," in *USENIX Sec.*, 2022, pp. 809–826.

[28] M. Hao, H. Li, H. Chen, P. Xing, G. Xu, and T. Zhang, "Iron: Private inference on transformers," in *NeurIPS*, 2022.

[29] W. Lu, Z. Huang, Z. Gu, J. Li, J. Liu, C. Hong, K. Ren, T. Wei, and W. Chen, "BumbleBee: Secure two-party inference framework for large transformers," in *NDSS*, 2025.

[30] S. Halevi and V. Shoup, "Algorithms in HElib," in *CRYPTO, Part I*, 2014, pp. 554–571.

[31] Q. Pang, J. Zhu, H. Möllering, W. Zheng, and T. Schneider, "BOLT: Privacy-preserving, accurate and efficient inference for transformers," in *S&P*, 2024, pp. 4753–4771.

[32] J. Bossuat, C. Mouchet, J. R. Troncoso-Pastoriza, and J. Hubaux, "Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys," in *EUROCRYPT, Part I*, 2021, pp. 587–617.

[33] J. Moon, D. Yoo, X. Jiang, and M. Kim, "THOR: Secure transformer inference with homomorphic encryption," in *CCS*, 2025, pp. 3765–3779.

[34] D. Park, E. Lee, and J. Lee, "Powerformer: Efficient and high-accuracy privacy-preserving language model with homomorphic encryption," in *ACL*, 2025, pp. 11 090–11 111.

[35] X. Jiang, M. Kim, K. E. Lauter, and Y. Song, "Secure outsourced matrix computation and application to neural networks," in *CCS*, 2018, pp. 1209–1222.

[36] L. Zhang, X. Wang, J. J. Sim, Z. Huang, J. Zhong, H. Wang, P. Duan, and K. Y. Lam, "MOAI: Module-optimizing architecture for non-interactive secure transformer inference," in *ICLR (Poster)*, 2026.

[37] T. Xu, L. Wu, R. Wang, and M. Li, "PrivCirNet: Efficient private inference via block circulant transformation," in *NeurIPS (Poster)*, 2024.

[38] E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl, "Efficient pseudorandom correlation generators: Silent OT extension and more," in *CRYPTO, Part III*, 2019, pp. 489–518.

[39] Z. Li, K. Yang, J. Tan, W. Lu, H. Wu, X. Wang, Y. Yu, D. Zhao, Y. Zheng, M. Guo, and J. Leng, "Nimbus: Secure and efficient two-party inference for transformers," in *NeurIPS (Poster)*, 2024.

[40] X. Gao, S. Fu, L. Liu, Z. Liu, Y. Luo, and Y. Wang, "Euston: Efficient and user-friendly secure transformer inference with non-interactivity," in *S&P*, 2026, pp. 1443–1462.

[41] J. Zhao and F. Wang, "Breaking Euston: Recovering private inputs from secure inference by exploiting subspace leakage," arXiv:2604.17238, 2026.

[42] T. Xu, W. Lu, J. Yu, Y. Chen, C. Lin, R. Wang, and M. Li, "Breaking the layer barrier: Remodeling private transformer inference with hybrid CKKS and MPC," in *USENIX Sec.*, 2025, pp. 2653–2672.

[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *NeurIPS*, 2019, pp. 8024–8035.

[44] Z. Peng, K. Xue, B. Zhu, Y. Huang, J. Li, M. Hao, and T. Zhang, "PrivTF: Efficient and privacy-preserving transformer inference with optimized protocols for secure NLP," *IEEE Trans. Netw. Sci. Eng.*, vol. 13, pp. 1887–1901, 2025.

[45] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "CrypTFlow2: Practical 2-party secure inference," in *CCS*, 2020, pp. 325–342.

[46] Y. Dong, W. Lu, Y. Zheng, H. Wu, D. Zhao, J. Tan, Z. Huang, C. Hong, T. Wei, and W. Chen, "PUMA: Secure inference of LLaMA-7B in five minutes," *Security and Safety*, vol. 4, p. 2025014, 2025.

[47] A. Y. L. Kei and S. S. M. Chow, "SHAFT: Secure, handy, accurate, and fast transformer inference," in *NDSS*, 2025.

[48] L. de Castro, D. Escudero, A. Agrawal, A. Polychroniadou, and M. Veloso, "EncryptedLLM: Privacy-preserving large language model inference via GPU-accelerated fully homomorphic encryption," in *ICML (Poster)*, 2025.

[49] W. Cho, G. Hanrot, T. Kim, M. Park, and D. Stehlé, "Fast and accurate homomorphic softmax evaluation," in *CCS*, 2024, pp. 4391–4404.

[50] D. Rathee, M. Rathee, R. K. K. Goli, D. Gupta, R. Sharma, N. Chandran, and A. Rastogi, "Sirnn: A math library for secure RNN inference," in *S&P*, 2021, pp. 1003–1020.

[51] K. Gupta, N. Jawalkar, A. Mukherjee, N. Chandran, D. Gupta, A. Panwar, and R. Sharma, "SIGMA: Secure GPT inference with function secret sharing," *Proc. Priv. Enhancing Technol.*, no. 4, pp. 61–79, 2024.

[52] K. Cheng, Y. Xia, A. Song, J. Fu, W. Qu, Y. Shen, and J. Zhang, "Mosformer: Maliciously secure three-party inference framework for large transformers," in *CCS*, 2025, pp. 4124–4138.

[53] G. Dessouky, F. Koushanfar, A. Sadeghi, T. Schneider, S. Zeitouni, and M. Zohner, "Pushing the communication barrier in secure computation using lookup tables," in *NDSS*, 2017.

[54] Z. Su, Q. Pang, S. Beyzerov, and W. Zheng, "FABLE: Batched evaluation on confidential lookup tables in 2PC," in *USENIX Sec.*, 2025, pp. 1945–1964.

[55] A. Y. L. Kei, L. K. L. Ng, J. P. K. Ma, and S. S. M. Chow, "Sort, Sweep, Mirror: Batch private interval lookup with logarithmic cost," in *S&P*, 2026, pp. 1937–1950.

[56] C. Zeng, D. He, Q. Feng, X. Yang, and Q. Luo, "SecureGPT: A framework for multi-party privacy-preserving transformer inference in GPT," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 9480–9493, 2024.

[57] Z. Xia, Q. Gu, W. Zhou, L. Xiong, J. Weng, and N. Xiong, "STR: secure computation on additive shares using the share-transform-reveal strategy," *IEEE Trans. Computers*, vol. 73, no. 2, pp. 340–352, 2024.

[58] A. H. Karp and P. W. Markstein, "High-precision division and square root," *ACM Trans. Math. Softw.*, vol. 23, no. 4, pp. 561–589, 1997.

[59] Y. Ding, H. Guo, Y. Guan, W. Liu, J. Huo, Z. Guan, and X. Zhang, "East: Efficient and accurate secure inference framework for transformer," *IEEE Transactions on Services Computing*, vol. 18, no. 4, pp. 2038–2046, 2025.

[60] L. Zhao and Z. Fang, "Secure outlier-aware large language model inference," in *ICLR (Poster)*, 2026.

[61] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "GPT3.int8(): 8-bit matrix multiplication for transformers at scale," in *NeurIPS*, 2022.

[62] Y. Ma, Y. Li, and S. Schmid, "FuseFSS: Efficient secure LLM inference with function secret sharing," in *ICML*, 2026.

[63] J. H. Cheon, D. Kim, and D. Kim, "Efficient homomorphic comparison methods with optimal complexity," in *ASIACRYPT, Part II*, 2020, pp. 221–256.

[64] Z. He, M. Wang, X. Gao, Y. Luo, L. Liu, and S. Fu, "ENSI: Efficient non-interactive secure inference for large language models," arXiv 2509.09424, 2025.

[65] D. Rho, T. Kim, M. Park, J. W. Kim, H. Chae, E. K. Ryu, and J. H. Cheon, "Encryption-friendly LLM architecture," in *ICLR (Poster)*, 2025.

[66] L. Rovida and A. Leporati, "Transformer-based language models and homomorphic encryption: An intersection with BERT-tiny," in *IWSPA@CODASPY*, 2024, pp. 3–13.

[67] T. Chen, H. Bao, S. Huang, L. Dong, B. Jiao, D. Jiang, H. Zhou, J. Li, and F. Wei, "THE-X: Privacy-preserving transformer inference with homomorphic encryption," in *ACL (Findings)*, 2022, pp. 3510–3520.

[68] Y. Zheng, Q. Zhang, S. S. M. Chow, Y. Peng, S. Tan, L. Li, and S. Yin, "Secure softmax/sigmoid for machine-learning computation," in *ACSAC*, 2023, pp. 463–476.

[69] D. Li, R. Shao, H. Wang, H. Guo, E. P. Xing, and H. Zhang, "MPCFormer: Fast, performant and private transformer inference with MPC," in *ICLR*, 2023.

[70] D. Chen, Y. Zhang, S. Kundu, C. Li, and P. A. Beerel, "RNA-ViT: Reduced-Dimension approximate normalized attention vision transformers for latency efficient private inference," in *ICCAD*, 2023, pp. 1–9.

[71] Y. Zhang, D. Chen, S. Kundu, C. Li, and P. A. Beerel, "SAL-ViT: Towards latency efficient private inference on ViT using selective attention search with a learnable softmax approximation," in *ICCV*, 2023, pp. 5116–5125.

[72] Y. Chen, Z. Tang, T. Lu, B. Zhang, Z. Shi, and Z. Wang, "Accelerating private large transformers inference through fine-grained collaborative computation," *IEEE Trans. Inf. Forensics Secur.*, vol. 20, pp. 7482–7497, 2025.

[73] J. Luo, Y. Zhang, Z. Zhang, J. Zhang, X. Mu, H. Wang, Y. Yu, and Z. Xu, "SecFormer: Fast and accurate privacy-preserving inference for transformer models via SMPC," in *ACL (Findings)*, 2024, pp. 13 333–13 348.

[74] X. Liu and Z. Liu, "LLMs can understand encrypted prompt: Towards privacy-computing friendly transformers," arXiv 2305.18396, 2023.

[75] H. Wu, W. Fang, Y. Zheng, J. Ma, J. Tan, and L. Wang, "Ditto: Quantization-aware secure inference of transformers upon MPC," in *ICML (Poster)*, 2024, pp. 53 346–53 365.

[76] J. Moon, Z. Omarov, D. Yoo, Y. An, and H. Chung, "Adaptive successive over-relaxation method for a faster iterative approximation of homomorphic operations," Cryptology ePrint 2024/1366, 2024.

[77] M. Zheng, Q. Lou, and L. Jiang, "Primer: Fast private transformer inference on encrypted data," in *DAC*, 2023, pp. 1–6.

[78] H. W. H. Wong, J. P. K. Ma, D. P. H. Wong, L. K. L. Ng, and S. S. M. Chow, "Learning DNN model with error - Exposing the hidden model of BAYHENN," in *IJCAI*, 2020, pp. 3529–3535.

[79] F. Zheng, C. Chen, Z. Han, and X. Zheng, "PermLLM: Private inference of large language models within 3 seconds under WAN," arXiv 2405.18744, 2024.

[80] M. Ugurbil, D. Mouris, M. B. Santos, J. Cabrero-Holgueras, M. de Vega, and S. Sengupta, "Fission: Distributed privacy-preserving large language model inference," in *CNS*, 2025, pp. 1–9.

[81] Z. Li, Y. Wang, K. Yang, Y. Yu, J. Gui, Y. Feng, N. Liu, M. Guo, and J. Leng, "On the (in-)security of the shuffling defense in the transformer secure inference," in *ACL*, 2026.

[82] W. Zeng, M. Li, W. Xiong, T. Tong, W. Lu, J. Tan, R. Wang, and R. Huang, "MPCViT: Searching for accurate and efficient MPC-friendly vision transformer with heterogeneous attention," in *ICCV*, 2023, pp. 5052–5063.

[83] S. Xia, W. Wang, Z. Wang, Y. Zhang, Y. Jin, D. Meng, and R. Hou, "CryptPEFT: Efficient and private neural network inference via parameter-efficient fine-tuning," in *NDSS*, 2026.

[84] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *ICLR (Poster)*, 2019.

[85] N. Dhyani, J. C. Mo, P. Yubeaton, M. Cho, A. Joshi, S. Garg, B. Reagen, and C. Hegde, "PriViT: Vision transformers for private inference," *Trans. Mach. Learn. Res.*, 2024.

[86] B. Zeng, Z. Pang, Y. Zhang, K. Zhao, T. Wu, G. Yang, L. Wang, and R. Wang, "PCFormer: Accelerating privacy-preserving transformer inference by partition and combination," in *AAAI*, 2026, pp. 28 076–28 084.

[87] I. Zimerman, M. Baruch, N. Drucker, G. Ezov, O. Soceanu, and L. Wolf, "Converting transformers to polynomial form for secure inference over homomorphic encryption," in *ICML (Poster)*, 2024, pp. 62 803–62 814.

[88] Y. Akimoto, K. Fukuchi, Y. Akimoto, and J. Sakuma, "Privformer: Privacy-preserving transformer with MPC," in *EuroS&P*, 2023, pp. 392–410.

[89] W. Wang and Y. Kuang, "CipherFormer: Efficient transformer private inference with low round complexity," in *CSCWD*, 2024, pp. 3054–3059.

[90] M. Wortsman, J. Lee, J. Gilmer, and S. Kornblith, "Replacing softmax with ReLU in vision transformers," arXiV 2309.08586, 2023.

[91] J. Ramapuram, F. Danieli, E. G. Dhekane, F. Weers, D. Busbridge, P. Ablin, T. Likhomanenko, J. Digani, Z. Gu, A. Shidani, and R. Webb, "Theory, analysis, and best practices for sigmoid self-attention," in *ICLR (Poster)*, 2025.

[92] Z. Pang, B. Feng, M. Luo, C. Liu, S. Xu, K. Zhao, Y. Wu, and B. Zeng, "Privacy-friendly adaptation of vision transformers for communication and latency-efficient private inference," in *WWW*, 2026, pp. 2695–2706.

[93] D. Rathee, D. Li, I. Stoica, H. Zhang, and R. A. Popa, "MPC-minimized secure LLM inference," arXiv 2408.03561, 2024.

[94] Z. Yao, R. Y. Aminabadi, M. Zhang, X. Wu, C. Li, and Y. He, "ZeroQuant: Efficient and affordable post-training quantization for large-scale transformers," in *NeurIPS*, 2022.

[95] T. Lu, B. Zhang, X. Zhang, and K. Ren, "A new PPML paradigm for quantized models," in *NDSS*, 2025.

[96] T. Lu, B. Zhang, L. Peng, B. Zheng, L. Li, and K. Ren, "Privacy-preserving inference for quantized BERT models," arXiv 2508.01636, 2025.

[97] S. Ma, H. Wang, L. Ma, L. Wang, W. Wang, S. Huang, L. Dong, R. Wang, J. Xue, and F. Wei, "The era of

1-bit LLMs: All large language models are in 1.58 bits," arXiv 2402.17764, 2024.

[98] Y. Zhang, J. Xue, M. Zheng, M. Xie, M. Zhang, L. Jiang, and Q. Lou, "CipherPrune: Efficient and scalable private transformer inference," in *ICLR (Poster)*, 2025.

[99] Y. Yu, Y. Zhou, Y. Chen, P. Soto, W. Xiong, and M. Li, "Cachemir: Fully homomorphic encrypted inference of generative large language model with KV cache," arXiv 2602.11470, 2026.

[100] W. Zeng, Y. Dong, J. Zhou, J. Tan, L. Wang, T. Wei, R. Wang, and M. Li, "MPCache: MPC-friendly KV cache eviction for efficient private LLM inference," in *NeurIPS (Poster)*, 2025.

[101] A. Liu, J. Liu, Z. Pan, Y. He, R. Haffari, and B. Zhuang, "MiniCache: KV cache compression in depth dimension for large language models," in *NeurIPS*, 2024.

[102] A. Fan, M. Lewis, and Y. N. Dauphin, "Hierarchical neural story generation," in *ACL*, 2018, pp. 889–898.

[103] T. Araki, J. Furukawa, K. Ohara, B. Pinkas, H. Rosemarin, and H. Tsuchida, "Secure graph analysis at scale," in *CCS*, 2021, pp. 610–629.

[104] Z. Liang, P. Wang, R. Zhang, N. Xu, S. Zhang, L. Xing, H. Bai, and Z. Zhou, "MERGE: Fast private text generation," in *AAAI*, 2024, pp. 19 884–19 892.

[105] Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," in *ICML*, 2023, pp. 19 274–19 286.

[106] Z. Li, Y. Guan, K. Yang, Y. Feng, N. Liu, Y. Yu, J. Leng, and M. Guo, "An efficient private GPT never autoregressively decodes," in *ICML (Poster)*, 2025.

[107] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *ICLR*, 2022.

[108] S. Chen, C. Ge, Z. Tong, J. Wang, Y. Song, J. Wang, and P. Luo, "AdaptFormer: Adapting vision transformers for scalable visual recognition," in *NeurIPS*, 2022.

[109] C. Riquelme, J. Puigcerver, B. Mustafa, M. Neumann, R. Jenatton, A. S. Pinto, D. Keysers, and N. Houlsby, "Scaling vision with sparse mixture of experts," in *NeurIPS*, 2021, pp. 8583–8595.

[110] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *J. Mach. Learn. Res.*, vol. 23, pp. 120:1–120:39, 2022.

[111] Y. Zhou, T. Xu, J. Hong, Y. Wu, and M. Li, "CryptoMoE: Privacy-preserving and scalable mixture of experts inference via balanced expert routing," in *NeurIPS (Poster)*, 2025.

[112] B. Shen, Y. Chen, P. Yang, B. Zhang, X. Zhang, and Z. L. Jiang, "SecMoE: Communication-efficient secure MoE inference via select-then-compute," in *AAAI*, 2026, pp. 25 286–25 294.

[113] J. Li, Y. Jiang, Y. Zhu, C. Wang, and H. Xu, "Accelerating distributed MoE training and inference with Lina," in *USENIX ATC*, 2023, pp. 945–959.

[114] Z. Liu, J. Wang, T. Dao, T. Zhou, B. Yuan, Z. Song, A. Shrivastava, C. Zhang, Y. Tian, C. Ré, and B. Chen, "Deja Vu: Contextual sparsity for efficient LLMs at inference time," in *ICML*, 2023, pp. 22 137–22 176.

[115] G. Yan, Y. Zhang, Z. Guo, L. Zhao, X. Chen, C. Wang, W. Wang, D. Meng, and R. Hou, "Comet: Accelerating private inference for large language model by predicting activation sparsity," in *S&P*, 2025, pp. 2827–2845.

[116] Y. Xue, L. Liu, Y. Luo, B. Sun, and S. Fu, "PrivMLLM: Efficient three-party multimodal large language model secure inference supported prompt privacy," in *ICNP*, 2025, pp. 1–24.

[117] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *ICLR*, 2019.

[118] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009, University of Toronto Technical Report.

[119] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," 2015, Stanford CS 231N.

[120] Y. Li, Y. Duan, Z. Huang, C. Hong, C. Zhang, and Y. Song, "Efficient 3PC for binary circuits with application to maliciously-secure DNN inference," in *USENIX Sec.*, 2023, pp. 5377–5394.

[121] K. Gupta, N. Chandran, D. Gupta, J. Katz, and R. Sharma, "SHARK: Actively secure inference using function secret sharing," in *S&P*, 2025, pp. 2472–2490.

[122] Y. Lv, R. Zhang, Z. Zhang, Z. Wan, L. Zhang, M. Xue, J. Li, and Y. Cao, "SMASH: Scalable maliciously secure hybrid multi-party computation framework for privacy-preserving large language models," in *USENIX Sec.*, 2026.

[123] L. de Castro, R. Agrawal, R. T. Yazicigil, A. P. Chandrakasan, V. Vaikuntanathan, C. Juvekar, and A. Joshi,

"Does fully homomorphic encryption need compute acceleration?" Cryptology ePrint 2021/1636, 2021.

[124] S. Jayashankar, J. Kim, M. B. Sullivan, W. Zheng, and D. Skarlatos, "A scalable multi-GPU framework for encrypted large-model inference," arXiv 2512.11269, 2025.

[125] Llama Team, "The llama 3 herd of models," arXiv 2407.21783, 2024.

[126] C. Lin, T. Xu, Z. Yang, R. Wang, R. Huang, and M. Li, "FastQuery: Communication-efficient embedding table query for private LLMs inference," in *DAC*, 2024, pp. 146:1–146:6.

[127] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *CRYPTO*, 2012, pp. 868–886.

[128] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," Cryptology ePrint 2012/144, 2012.

[129] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *ASIACRYPT, Part I*, 2017, pp. 409–437.

[130] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "A full RNS variant of approximate homomorphic encryption," in *SAC*, 2018, pp. 347–368.

[131] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, "More efficient oblivious transfer and extensions for faster secure computation," in *CCS*, 2013.

[132] I. Turc, M. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: The impact of student initialization on knowledge distillation," arXiv 1908.08962, 2019.

[133] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019, OpenAI report.

[134] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. Canton-Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open foundation and fine-tuned chat models," arXiv 2307.09288, 2023.

[135] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," in *ICLR (Poster)*. OpenReview.net, 2017.

[136] D. Paperno, G. Kruszewski, A. Lazaridou, Q. N. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernández, "The LAMBADA dataset: Word prediction requiring a broad discourse context," in *ACL*, 2016, pp. 1525–1534.

[137] N. Chandran, D. Gupta, A. Rastogi, R. Sharma, and S. Tripathi, "EzPC: Programmable and efficient secure two-party computation for machine learning," in *EuroS&P*, 2019, pp. 496–511.

[138] J. Ma, Y. Zheng, J. Feng, D. Zhao, H. Wu, W. Fang, J. Tan, C. Yu, B. Zhang, and L. Wang, "SecretFlow-SPU: A performant and user-friendly framework for privacy-preserving machine learning," in *USENIX ATC*, 2023, pp. 17–33.

[139] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system for large-scale machine learning," in *USENIX OSDI*, 2016, pp. 265–283.

[140] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *EMNLP (Demos)*, 2020, pp. 38–45.

A Consolidated Research Agenda

The main text states each open problem and outlook alongside its motivating analysis. Table 2 consolidates the 12 open problems and 16 outlooks by theme and representative systems.

B Literature-Selection Methodology

Within the frozen public-release window (1 January 2022 to 30 June 2026), we build the corpus in three stages: base census, eligibility screening, and auditable supplementation. These stages separately determine how candidates are discovered, admitted to the corpus, and subsequently emphasized in the analysis. The counted corpus is deliberately scoped to a frozen publication universe rather than every public work that mentions cryptographic transformer inference.

B.1 Base Census and Publication Universe

The base census uses a high-recall search over predefined sources, venues, and keywords. Its primary venue set is restricted to a frozen, named list of peer-reviewed conferences and journals spanning security, privacy, and cryptography; AI, ML, computer vision, and natural language processing; and systems, architecture, and design automation. The exact named list is recorded in the audit artifact together with the frozen sources, executed queries, and candidate ledger. Peer-reviewed papers outside this venue set are treated as contextual references rather than corpus entries. To account for publication lag, technically mature pure preprints may enter through a separate frontier-supplement process.

Searches combine terms from three groups: (1) models ("transformer," "BERT," "GPT," "LLM," "large language model," "ViT," and "vision transformer"), (2) private inference terminology ("private inference," "privacy-preserving inference," "secure inference," "encrypted inference," and "oblivious inference"), and (3) cryptographic backends ("homomorphic encryption," "HE," "MPC," "secure multiparty computation," "secret sharing," "FSS," "function secret sharing," "garbled circuit," and "oblivious transfer"). Searches do not require terms from all three groups simultaneously because relevant works often use different terminology.

B.2 Uniform Eligibility Screening

For every candidate work w, corpus membership is determined by the same predicate, $E(w) = C(w) \wedge R(w) \wedge A(w) \wedge F(w) \wedge T(w) \wedge S(w)$. Existing rows receive no grandfathered exemption from this predicate. The acronym *CRAFTS* summarizes the six admission requirements.

C—**Cryptographic privacy.** The work performs private inference with cryptographic computation, such as SS, HE, OT, GC, FSS, or their combinations. Non-cryptographic approaches (*e.g.*, TEE, differential privacy, federated learning), verifiable-only systems, and orthogonal attack studies (*e.g.*, membership inference, model extraction) are out of scope.

R—**Recognized publication venue.** The work must either be peer reviewed in the frozen primary venue set or qualify as a pure preprint under the frontier-supplement rule below. Peer-reviewed works outside the primary venue set may be discussed as contextual but are not counted corpus entries.

A—**Analyzability.** Public technical detail must suffice to identify the private computation, deployment or threat model, key techniques, and taxonomy placement.

F—**Integrated and materially distinguishable private inference framework.** The work must contribute a materially distinct protocol, cryptographic representation, cryptography–ML architecture, or execution mechanism and integrate it into private inference. Component-only contributions, reproductions, parameter variants, and implementation refinements are treated as contextual. For example, FastQuery [126] is contextual as it contributes an isolated embedding-table component rather than a complete private inference framework.

T—**Substantive transformer instantiation.** The framework must be instantiated or evaluated end-to-end on a transformer workload (*e.g.*, BERT, GPT, or ViT). Techniques originating elsewhere remain eligible if their transformer instantiation materially affects the design or evaluation of private inference; for example, PPML-Quantized [95] and SHARK [121].

S—**Selection window, versioning, and deduplication.** The work must be publicly released within the frozen window and represent a distinct framework counted once. We use the latest public version for technical classification and the earliest public release for chronology and ordering in Tables 6–7.

B.3 Auditable Supplementation

We improve recall after the base census through citation tracing, existing SoKs, and technically relevant public preprints. Chen *et al.*'s 61-work SoK [14] serves as an independent cross-check, but not an eligibility authority. A pure preprint may enter through the frontier-supplement route only if it satisfies *CRAFTS*, provides sufficient technical detail for evaluation, and contributes a materially distinct and analytically relevant design not already represented in the corpus. Each screened preprint receives an audit reason code (inclusion, component-only, duplicate, insufficient-maturity/evidence, no-distinct-design-point, or out-of-scope). Peer-reviewed works outside the frozen primary venue set remain contextual.

Audit trail and analytical emphasis. For every candidate, the audit record logs the discovery route, publication-route decision, eligibility rationale, version relationship, and deduplication outcome. This makes the process auditable and reproducible within the declared search universe without claiming exhaustive coverage. Analytical emphasis is assigned after inclusion according to a framework's role in the taxonomy and is not an additional admission criterion.

Table 2: Consolidated Research Agenda: Open Problems (OPs) and Outlooks (OLs) by Theme

Theme	Core Question	Items	Representative Systems	Example Directions
Protocol Efficiency	How can transformer primitives be made cheaper while preserving privacy and accuracy?	OP2–6; OL1–2, 4–6	BOLT [31], Nimbus [39], THOR [33], MOAI [36], SIGMA [51], SHAFT [47]	Compiler-synthesized packing automation; batch-LUT-based nonlinear evaluations [54, 55]
Cross-Level Co-Design	How can systems, models, and cryptography be co-designed to maximize end-to-end efficiency?	OP1, 7; OL3, 7, 8, 11, 12	CipherGPT [12], NEXUS [13], BLB [42], PPML-Quantized [95], CryptPEFT [83]	Phase-aware cross-step amortization; joint approximation/fusion planning; efficient private dynamic quantization; private adaptation co-design; cryptography-aware decoding
Dynamic Execution	How can private inference leverage data-dependent execution without leaking decisions?	OP9–10; OL9–10	MPCache [100], SecMoE [112], Cachemir [99],	Oblivious or bounded adaptive execution; distributed MoE across independent servers; lightweight utility predictors for early token pruning
Security and Leakage	How should privacy leakage from structural disclosures and public auxiliary artifacts be defined and measured?	OP8; OL15	CipherPrune [98], Post [106], Comet [115]	Leakage accounting for retention/sparsity disclosures and draft-model exposure
Malicious Security	How can active security survive preprocessing, nonlinear evaluation, and representation changes end to end?	OL16	SHARK [121], Privformer [88], Mosformer [52], SMASH [122]	Authenticated/interactive FSS; VDPFs; lightweight ZKPoKs; malicious share conversion and proof-matched preprocessing
Evaluation, Robustness, and Scalability	How can private inference be performed and evaluated reliably at realistic tasks and model scales?	OP11, 12; OL13, 14	All surveyed systems	Standardized evaluation benchmarks accounting for retraining and generalization cost

C Transformer Architecture

A transformer processes inputs through stacked blocks, each combining linear projections with nonlinear activations.

Embedding Layer. Token embedding maps input token i to $\mathbf{x} = \mathcal{T}[i] \in \mathbb{R}^d$ via a lookup table $\mathcal{T}$ of size v; encoding the token as a one-hot vector $\mathbf{e}_i \in \{0,1\}^v$ reduces this to a matrix-vector product $\mathbf{e}_i \mathbf{W}_{\mathcal{T}}$, the entry point for secure linear protocols (§3.3). Vision transformers (*e.g.*, ViT [20]) replace token embeddings with patch embeddings, projecting flattened image patches to d dimensions via a strided convolution [15].

Multi-Head Attention (MHA). An attention layer maps query $\mathbf{Q}$, key $\mathbf{K}$, and value $\mathbf{V} \in \mathbb{R}^{m \times d}$—obtained by linear projections of input $\mathbf{X} \in \mathbb{R}^{m \times d}$ with sequence length m—to

$$\mathsf{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathsf{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}.$$

MHA computes H attention heads in parallel and concatenates their outputs: $\mathsf{MHA}(\mathbf{X}) = (\mathsf{head}_0 \| \cdots \| \mathsf{head}_{H-1})\mathbf{W}^O$,

$$\mathsf{head}_i = \mathsf{Attention}(\mathbf{X}\mathbf{W}_i^Q, \mathbf{X}\mathbf{W}_i^K, \mathbf{X}\mathbf{W}_i^V) \quad \text{for } i \in [H],$$

where $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{d \times (d/H)}$ are per-head projection matrices and $\mathbf{W}^O \in \mathbb{R}^{d \times d}$ is the output projection. The products $\mathbf{Q}_i \mathbf{K}_i^\top$ and $\mathsf{Softmax}(\cdot)\mathbf{V}_i$ involve two encrypted matrices, creating the ct-ct multiplications that do not arise in CNNs (§3).

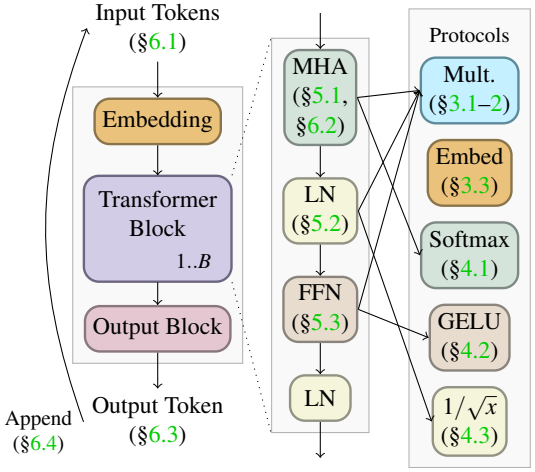

Figure 1: Transformer architecture and subprotocols: Inputs are mapped to embeddings, processed by B stacked blocks (LN = LayerNorm), and passed to an output block. The append step applies to generative models only.

Feed-Forward Network (FFN). An FFN applies two linear layers separated by a nonlinear activation F:

$$\mathsf{FFN}(\mathbf{X}) = F(\mathbf{X}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2,$$

where $\mathbf{W}_1 \in \mathbb{R}^{d \times d_{\text{FFN}}}$, $\mathbf{b}_1 \in \mathbb{R}^{d_{\text{FFN}}}$, $\mathbf{W}_2 \in \mathbb{R}^{d_{\text{FFN}} \times d}$, $\mathbf{b}_2 \in \mathbb{R}^d$. In typical modern transformers, $F = \mathsf{GELU}$, where

$$\mathsf{GELU}(x) = 0.5x\left(1 + \mathsf{Erf}\left(\tfrac{x}{\sqrt{2}}\right)\right), \quad \mathsf{Erf}(x) = \tfrac{2}{\sqrt{\pi}} \int_0^x e^{-u^2}\, du.$$

Alternative activation functions are discussed in §5.3.

Layer Normalization (LayerNorm). LayerNorm normalizes each token vector $\mathbf{x} \in \mathbb{R}^d$ by its mean μ and standard deviation σ, then applies a learned affine rescaling $(\mathbf{g}, \mathbf{b})$:

$$\mathsf{LayerNorm}(\mathbf{x}) = \frac{\mathbf{g}}{\sigma} * (\mathbf{x} - \mu) + \mathbf{b},$$

$$\mu = \frac{1}{d}\sum_{i=1}^{d} \mathbf{x}[i], \qquad \sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d} (\mathbf{x}[i] - \mu)^2 + \varepsilon},$$

where $*$ denotes element-wise product and ε is a small constant added for numerical stability. The only non-polynomial bottleneck is the division by σ (§4.3).

Transformer Block. Traditional transformers apply residual connections and LayerNorm around both MHA and FFN (post-norm). A block with input $\mathbf{X} \in \mathbb{R}^{m \times d}$ computes:

$$\mathbf{Y} = \mathsf{LayerNorm}(\mathbf{X} + \mathsf{MHA}(\mathbf{X})),$$

$$\mathsf{Block}(\mathbf{X}) = \mathsf{LayerNorm}(\mathbf{Y} + \mathsf{FFN}(\mathbf{Y})).$$

D Cryptographic Primitives and Backends

A (t,n)-SS scheme distributes a secret $x \in \mathbb{Z}_p$ into n shares such that only sets of $\geq t$ shares can reconstruct x. We primarily use $(2,2)$ additive SS, denoted $[\![x]\!]$, where $[\![x]\!]_0 + [\![x]\!]_1 = x \bmod p$. *Replicated secret sharing* (RSS) distributes overlapping shares across multiple parties; the common $(2,3)$-RSS setting, denoted $\langle x \rangle$, provides efficient multiplication and lightweight malicious-security checks through redundancy.

HE allows computation directly on ciphertexts $[x]$. We focus on *leveled HE (LHE)*, the paradigm adopted by most HE-based frameworks. LHE fixes at setup a supported *multiplicative depth*, with public parameters chosen accordingly; deeper computation requires larger parameters or ciphertext refresh through *bootstrapping*.

Leading HE schemes BFV [127,128] and CKKS [129,130] are ring-LWE-based and support *single-instruction multiple-data (SIMD) batching*, though ciphertext rotations in matrix products remain a bottleneck. BFV supports exact modular arithmetic, making it naturally compatible with arithmetic SS. CKKS encodes approximate reals, which fits ML workloads well but complicates conversion to exact arithmetic shares.

Table 3: Commonly Supported Settings by Backends

Backend		Security			Deployment		
	Example	2PC	(2+1)PC	3PC	Obliv.	Outs.	NI
Hybrid HE-MPC	[31]	●	○	○	●	○	○
OT-Based MPC	[82]	●	○	○	●	●	○
Dealer-Aided MPC	[47]	○	●	○	●	●	○
Pure-SS	[51]	●	●	●	●	●	○
Pure-HE	[33]	●	○	○	●	○	●

Obliv. = Oblivious; Outs. = Outsourced; NI = Noninteractive.

Oblivious transfer (OT) allows a receiver to obtain one of two sender-held messages without revealing its choice; *OT extension* [38,131] makes large-scale OT practical using mainly symmetric-key operations. *Garbled circuits (GC)* provide constant-round 2PC, making them effective for nonlinear functions but expensive for large arithmetic workloads. *Function secret sharing (FSS)* splits a function into secret shares that can be evaluated independently and combined afterward; *Distributed point functions (DPFs)* are a common special case used for efficient lookup-based nonlinear evaluation.

Cryptographic backend is the combination of primitives underlying layer evaluation. It determines a system's performance, security model, and deployment scenario.

Hybrid HE-MPC systems combine SS with HE and OT/GC, typically using HE for linear layers and OT/GC for nonlinearities. *OT-based MPC* and *dealer-aided MPC* replace HE-based linear evaluation with OT- or dealer-generated correlated randomness, while using OT/GC for nonlinearities. *Pure-SS* systems rely only on SS variants, either approximating nonlinearities with low-degree polynomials or evaluating them through FSS-based lookup tables, often in (2+1)PC or honest-majority 3PC settings. OT-based MPC has lower computation but higher communication compared to hybrid HE-MPC. Dealer-aided MPC and pure SS achieve fast online execution at the cost of additional non-colluding assumptions. *Pure-HE* systems perform inference entirely on encrypted data (*e.g.*, under CKKS), requiring neither interaction nor additional parties, but incur substantial costs from rotations, bootstrapping, and nonlinear-function approximations.

No backend is uniformly best [15]. Hybrid HE-MPC trades conversion overhead for flexibility; dealer-aided MPC and pure SS offer fast online execution under additional trust assumptions; pure HE offers noninteraction at high computational cost. Table 3 summarizes representative deployment settings; Table 6 gives the full framework classification.

Table 4: Security Assessment of Identified Concerns

Approach	Mechanism and Security Concern	Implication	Evidence
Intermediate Reveal [67]	Intermediate activations are decrypted and processed in plaintext, revealing information beyond the output-only baseline.	Exposed activations may reveal private inputs or model parameters, enabling hidden-model recovery attacks [78].	**Established (structural)**: plaintext disclosure directly exceeds the output-only baseline.
Shuffling Defenses [79,80]	Activations are disclosed after shuffling, which hides positions but not values or distributions.	PermLLM-style shuffling can expose model information, while Fission's split-and-shuffle can retain distribution information.	**Established (empirical)**: recent work [81] reports model extraction against shuffling; Fission [80] reports residual distribution leakage.
Masked Denominator [56]	Private division reconstructs the masked denominator after fixed-point multiplication. In $\mathbb{Z}_{2^{64}}$ with $f = 13$, equal-output inputs $(2^f, 2^f)$ and $(2^{f+1}, 2^{f+1})$ yield openings distinguishable by parity even under uniform masking.	The opened value is not simulatable from the quotient alone and leaks a denominator property; this is a protocol-level finding, not an end-to-end recovery claim.	**Established (structural)**: the equal-output witness invalidates the uniform-product premise used by the division security argument [56].
SVD Masking [40]	Euston encrypts only the singular values but leaves singular vectors exposed, which may expose the masking matrix's subspace structure.	An attack exploits this leakage to separate masks and reconstruct private inputs under the semi-honest model.	**Preliminary**: reported in a public preprint [41]; independent validation remains unresolved.

E Additional Technical Details

E.1 HE Packing Case Studies

BOLT [31] departs from HE-to-SS attention by introducing efficient HE-based ct-ct matrix multiplication. For pt-ct, it combines column-packed ciphertexts with diagonally packed weights, fully utilizing slots and avoiding coefficient-encoding waste at the cost of additional rotations; BSGS [32] reduces those rotations by 2.33–9.33×. For ct-ct, BOLT targets the two packing configurations that naturally arise in attention: column-packed by row-packed multiplication is evaluated as a sum of outer products, while diagonal-packed by column-packed multiplication exploits the square attention structure to rotate and mask without ciphertext repacking.

THOR [33] unifies pt-ct and ct-ct with a single diagonal-major encoding. The common format eliminates inter-layer repacking, supports efficient transposition, enables SIMD parallelism across multiple matrices, and reduces costs through blockwise processing and BSGS-based diagonal-summation.

Powerformer [34] builds on row packing [35]. For pt-ct, CKKS complex-number packing merges two real-valued ciphertexts into one, halving ciphertexts and rotations. For ct-ct, it develops a column-packing variant and a square-matrix optimization, later generalized to rectangular matrices. These reduce key switchings by 35%–91% over Jiang *et al.* [35].

MOAI [36] co-designs packing with the transformer computation graph. Compatible formats for pt-ct projections and ct-ct attention eliminate inter-layer repacking; interleaved batching removes internal rotations in ct-ct multiplication, while rotation-free softmax and LayerNorm further reduce latency relative to THOR and Powerformer.

Table 5: Architecture Hyperparameters of Common Models

Model	# of Params	B	H	d	d_{FFN}	v
BERT-tiny [132]	4M	2	2	128	512	30522
BERT-base [19]	110M	12	12	768	3072	30522
BERT-large [19]	340M	24	16	1024	4096	30522
GPT-2-base [133]	117M	12	12	768	3072	50257
LLaMA2-7B [134]	7B	32	32	4096	11008	32000
LLaMA3-8B [125]	8B	32	32	4096	14336	128256
ViT-base [20]	86.6M	12	12	768	3072	-

B = Number (#) of blocks; H = # of attention heads; d = Embedding dimension; d_{FFN} = FFN intermediate dimension; v = Vocabulary size.

E.2 Client-Side Linear Outsourcing

HE-based inference normally assigns homomorphic computation to the server. Nimbus [39] inverts this assignment for linear layers: the server preprocesses and encrypts weights row-wise, allowing the client to multiply activations with encrypted weight rows online. This avoids rotation-heavy server-side inner products and yields 2.9–12.5× linear-layer speedups over prior work [29], while increasing client cryptographic work proportional to the weight-matrix dimension.

F Common Models and Benchmarks

From a computational perspective, most transformers share a common structure (§C). Their primary differences lie in the architectural hyperparameters summarized in Table 5.

In natural language processing, a widely used benchmark is the general language understanding evaluation (GLUE) benchmark [117]. GLUE comprises a diverse set of natural

language understanding tasks, including:

SST-2. A binary sentiment classification task on single sentences from movie reviews (positive vs. negative).

QNLI. A binary classification task reformulated from a question answering dataset, determining whether a sentence contains the answer to a given question.

MRPC. A binary classification task on sentence pairs from online news sources to identify semantic equivalence.

RTE. A small binary classification dataset for classifying sentence pairs into either entailment or non-entailment.

CoLA. A binary single-sentence classification task to determine grammatical acceptability for English sentences.

MNLI. A classification task similar to RTE, but with substantially more training data for classifying sentence pairs into three classes: entailment, contradiction, or neutral.

STS-B. A regression task predicting semantic similarity between two sentences on a scale from 0 to 5.

QQP. A binary classification task to detect whether two Quora questions are semantically equivalent.

Generation and language modeling benchmarks include:

WikiText-103 [135]. A large-scale language modeling benchmark constructed from Wikipedia articles. The objective is to predict future tokens in long-form text.

LAMBADA [136]. A language modeling benchmark designed to evaluate long-range contextual reasoning. Successful prediction requires integrating information across broader preceding context rather than relying on local context.

In computer vision, common benchmarks include:

CIFAR-10/100 [118]. Widely used benchmark datasets for image classification. They contain 10/100 classes of animals and vehicles. Each sample is a 32×32 RGB color image.

Tiny-ImageNet [119]. A scaled-down ImageNet benchmark for image classification. It contains 120000 RGB images of size 64×64 across 200 classes, serving as a middle ground between the simpler CIFAR-10/100 and the full ImageNet.

G Deployment Tooling

Private transformer frameworks typically build on generic secure-computation libraries that impose steep usability barriers. EzPC [137] demands deep cryptographic expertise, SPU [138] departs from standard PyTorch [43] and TensorFlow [139] workflows, and CrypTen [25] lacks essential transformer modules such as embeddings. SHAFT [47] mitigates the last gap by extending CrypTen with Hugging Face [140] compatibility, but remains limited to SS-based execution. HE-based frameworks still require manual, model-specific packing and rotation decisions. A compiler that automates these choices from a standard computation graph has yet to appear.

Table 6: Private Transformer Framework Summary

	Framework	Basic Info			Property			Crypto Tool					Optim			Linear				Nonlinear						Modification			
		Ref	Venue	Year	Sec	Acc	Open	SS	HE	OT	GC	FSS	Bw	Prep	GPU	MPC	Pt-Ct	Ct-Ct	Emb	Repl	Poly	Iter	LUT	FS	Emp	Attn	LN	Act	Train
2PC Interactive	THE-X	[67]	AI	22	S!	◐	○	○	●	○	○	○	-	○	○	○	◐	◐	○	●	○	○	○	○	○	○	●	○	●
	MPCViT	[82]	AI*	22	S	-	◐	●	○	●	●	○	-	○	○	◐	○	○	-	●	○	◐	○	○	○	●	○	●	●
	Iron	[28]	AI*	22	S	●	○	●	●	●	○	○	-	○	○	◐	●	○	○	○	○	◐	◐	○	○	○	○	○	○
	Primer	[77]	Sys*	23	S	-	○	●	●	●	●	○	-	●	○	○	◐	●	○	○	○	○	○	○	○	○	◐	●	○
	Liu–Liu	[74]	Pre	23	S	●	●	●	●	●	○	○	-	●	○	◐	◐	○	○	◐	○	○	○	○	○	○	●	○	○
	CipherGPT	[12]	Sec	23	S	○	○	●	●	●	○	○	M	●	○	◐	●	○	●	○	○	◐	●	○	○	○	○	○	○
	East	[59]	Sys	23	S	●	○	●	●	●	○	○	-	●	○	◐	◐	○	○	○	○	◐	●	○	○	○	○	○	○
	PriViT	[85]	AI	23	S	○	○	●	●	●	●	○	-	●	○	◐	◐	○	○	●	○	◐	○	○	○	○	●	○	◐	●
	BumbleBee	[29]	Sec*	23	S	●	●	●	●	●	○	○	M	○	○	◐	●	○	○	○	◐	◐	○	○	○	○	○	○	○
	BOLT	[31]	Sec*	23	S	●	●	●	●	●	○	○	-	○	○	○	●	●	○	○	●	◐	○	○	●	○	○	○	◐
	CipherFormer	[89]	Sys	24	S	-	○	●	●	●	●	○	M	●	○	◐	○	○	○	○	○	○	○	○	○	○	●	○	◐	○
	PPML-Quantized	[95]	Sec*	24	S	○	○	●	○	●	○	○	Q	●	●	●	○	○	○	○	○	○	●	○	○	○	○	○	◐	○
	SecureGPT	[56]	Sec	24	S!	○	○	●	○	○	○	○	-	○	○	◐	○	○	○	○	○	●	○	○	○	○	○	○	○	○
	Nimbus	[39]	AI	24	S	●	●	●	●	●	○	○	-	●	○	◐	●	○	○	○	●	◐	○	○	●	○	○	○	◐
	PrivCirNet	[37]	AI	24	S	○	●	●	●	●	○	○	-	○	○	○	●	◐	○	-	-	-	-	-	-	○	○	○	○
	FastLMPI	[72]	Sec	24	S	○	○	●	●	●	○	○	-	○	○	◐	◐	○	○	○	●	◐	◐	○	○	○	○	○	○
	MPCache	[100]	AI	25	S	●	●	●	●	●	○	○	-	○	●	◐	◐	○	○	-	-	-	-	-	-	○	○	○	○
	CipherPrune	[98]	AI	25	S	-	●	●	●	●	○	○	-	○	○	◐	◐	○	○	-	-	-	-	-	-	○	○	○	●
	Post	[106]	AI	25	S	-	○	●	●	●	○	○	-	○	○	◐	◐	○	○	-	-	-	-	-	-	○	○	○	●
	BLB	[42]	Sec*	25	S	●	●	●	●	●	○	○	M	●	○	○	◐	●	○	○	◐	◐	○	○	○	○	○	○	○
	PrivTF	[44]	Sys	25	S	●	○	●	○	●	○	○	M	○	○	◐	○	○	◐	○	○	◐	○	○	○	○	●	○	○
	CryptoMoE	[111]	AI	25	S	●	●	●	●	●	○	○	-	○	○	◐	◐	○	○	-	-	-	-	-	-	○	○	○	○
	SecMoE	[112]	AI*	26	S	●	○	●	●	●	○	○	-	○	○	◐	◐	○	○	○	◐	◐	○	○	○	○	○	○	○
	PCFormer	[86]	AI*	26	S	-	○	●	●	○	○	○	-	○	●	○	◐	◐	○	-	-	-	-	-	-	●	○	●	●
	SecViT	[92]	Sys*	26	S	●	○	●	○	●	●	○	-	○	○	◐	○	○	-	●	○	○	○	○	○	●	○	◐	●
	SMASH	[122]	Sec*	26	M	○	●	●	●	○	○	○	-	●	○	◐	◐	○	○	○	○	○	●	○	○	○	○	○	○
2PC Noninteractive	PolyTransformer	[87]	AI	23	S	◐	○	○	●	○	○	○	-	○	○	○	◐	◐	○	●	●	○	○	○	●	○	●	○	◐
	NEXUS	[13]	Sec*	24	S	●	◐	○	●	○	○	○	-	●	●	○	◐	●	○	○	●	◐	○	○	●	○	○	○	○
	Rovida–Leporati	[66]	Sec	24	S	◐	●	○	●	○	○	○	-	○	○	○	◐	◐	○	○	●	○	○	○	○	○	◐	○	○
	Powerformer	[34]	AI*	24	S	●	●	○	●	○	○	○	-	○	●	○	●	●	○	●	●	○	○	○	●	○	◐	○	●
	Rho *et al.*	[65]	AI	24	S	●	●	○	●	○	○	○	-	○	●	○	◐	◐	○	○	○	○	○	○	○	●	○	●	●
	THOR	[33]	Sec*	24	S	●	○	○	●	○	○	○	-	○	●	○	●	●	○	○	●	◐	○	○	◐	○	○	○	○
	EncryptedLLM	[48]	AI	25	S	●	●	○	●	○	○	○	-	○	●	○	◐	◐	○	○	◐	◐	○	○	●	○	○	○	○
	MOAI	[36]	AI	25	S	●	●	○	●	○	○	○	-	○	●	○	◐	●	○	○	●	◐	○	○	◐	○	○	○	○
	ENSI	[64]	Pre	25	S	◐	◐	○	●	○	○	○	T	○	●	○	◐	◐	○	○	◐	○	○	○	○	●	○	●	○
	Cachemir	[99]	Pre	26	S	●	○	○	●	○	○	○	-	○	●	○	●	◐	○	○	◐	◐	○	○	◐	○	○	○	○
	Euston	[40]	Sec*	26	S?	●	◐	○	●	○	○	○	-	●	●	○	●	◐	○	○	●	◐	○	○	◐	○	○	○	○
(2+1)PC	MPCFormer	[69]	AI*	22	S	◐	●	●	○	○	○	○	-	●	●	◐	○	○	○	●	○	○	○	○	●	○	●	○	◐
	RNA-ViT	[70]	Sys	23	S	●	○	●	○	●	○	○	-	●	○	◐	○	○	-	●	○	◐	○	○	●	○	○	●	◐
	MERGE	[104]	AI*	23	S	○	●	●	○	○	○	○	-	●	●	◐	○	○	○	-	-	-	-	-	-	○	◐	○	◐
	SIGMA	[51]	Sec	23	S	●	●	●	○	○	○	●	M	●	●	◐	○	○	○	○	○	○	●	○	○	○	○	○	○
	SAL-ViT	[71]	AI*	23	S	●	○	●	○	●	○	○	-	●	○	◐	○	○	-	●	○	◐	○	○	○	○	◐	○	◐
	SecFormer	[73]	AI	24	S	◐	●	●	○	●	○	○	-	●	○	◐	○	○	○	◐	○	◐	○	●	●	○	○	○	◐
	PermLLM	[79]	Pre	24	S!	○	○	●	○	○	○	○	-	●	●	◐	○	○	○	○	○	○	○	○	○	○	○	○	○
	Marill	[93]	Pre	24	S	-	○	●	○	●	○	○	-	●	●	◐	○	○	○	-	-	-	-	-	-	●	○	○	○
	SHAFT	[47]	Sec*	25	S	●	●	●	○	●	○	○	-	●	●	◐	○	○	●	○	○	●	○	●	●	○	○	○	○
	Fission	[80]	Sec	25	S!	●	●	●	○	○	○	○	-	●	●	◐	○	○	○	○	○	○	○	○	○	○	○	○	○
	SHARK	[121]	Sec*	25	M	○	◐	●	○	○	○	●	M	●	○	◐	○	○	○	○	○	○	●	○	○	○	○	○	○
	Comet	[115]	Sec*	25	S	◐	○	●	○	●	○	○	-	●	●	●	○	○	○	○	○	◐	○	○	○	○	○	◐	●
	CryptPEFT	[83]	Sec*	25	S	●	●	●	○	●	○	○	-	●	●	◐	○	○	-	-	-	-	-	-	-	◐	◐	○	◐
	SOAL	[60]	AI	26	S	●	○	●	○	●	○	○	-	●	●	◐	○	○	○	○	○	●	◐	○	●	○	○	◐	●
	FuseFSS	[62]	AI*	26	S	●	○	●	○	○	○	●	M	●	●	◐	○	○	○	○	○	○	●	○	○	○	○	○	○
3PC	PUMA	[46]	Sec	23	S	●	●	●	○	○	○	○	-	○	○	◐	○	○	●	○	●	◐	○	○	○	○	○	○	○
	Privformer	[88]	Sec	23	M	○	○	●	○	○	○	○	-	○	○	◐	○	○	○	○	○	○	○	○	○	●	○	◐	○
	Ditto	[75]	AI	24	S	◐	●	●	○	○	○	○	Q	○	○	◐	○	○	○	◐	○	◐	○	○	○	○	○	○	◐
	Lu *et al.*	[96]	Pre	25	S	○	○	●	○	○	○	○	Q	●	○	◐	○	○	○	○	○	○	◐	○	○	○	○	◐	○
	Mosformer	[52]	Sec*	25	M	●	●	●	○	○	○	●	M	●	○	◐	○	○	○	○	○	○	●	○	●	○	○	○	○
	PrivMLLM	[116]	Sys	25	S	◐	○	●	○	○	○	●	-	●	●	◐	○	○	◐	○	○	○	◐	○	○	○	○	○	○

Works are ordered by release date (earliest version including preprint). Venue: Sec = Security; Sys = System; Pre = Preprint; * = "Top-tier" (ICORE A*) conference (poster/findings excluded). Sec: S = Semi-honest security; M = Malicious security; **?** = Preliminary issue; **!** = Established issue; The markers are orthogonal to the S/M adversary-model classification; see Table 4. Accuracy (Acc) is a within-work fidelity indicator relative to each framework's own plaintext baseline on its reported models/tasks: ● = Avg. loss $\leq 1\%$; ◐ = Loss $\leq 3\%$; ○ = Loss $> 3\%$, or not reported; - = No comparison with plaintext inference. Identical Acc symbols are not comparable across rows because baselines, models, tasks, and reported subsets differ; Table 7 records evaluation coverage. Openness (Open): ● = Open-source end-to-end implementation available; ◐ = Open-source but not end-to-end; ○ = Not open-source. Crypto tools: SS = Additive SS (all settings)/RSS (3PC); HE = Homomorphic encryption; OT = Oblivious transfer; GC = Garbled circuit; FSS = Function SS. Optimization: Bw = Bitwidth (-: Fixed, M: Mixed, Q: Quantized, T: Ternary/1.58-bit); Prep = Preprocessing. Technique usage: ● = New techniques/approaches; ◐ = Existing ones. ●/◐ indicates techniques with architectural modification (§5) or potential generalization/deployment issues (§8.1). MPC = Uses MPC for share-share/ciphertext-ciphertext (Ct-Ct) matrix multiplication; Pt-Ct = Plaintext-ciphertext multiplication; Ct-Ct = Ct-Ct multiplication; Emb = Native embedding (index input); Repl = Replacement; Poly = Polynomial (composite for pure-HE, piecewise otherwise); Iter = Iterative; LUT = Lookup table; FS = Fourier series; Emp = Empirically-tuned or distribution-specific; "-" in "Nonlinear" = Not discussed. (Required) Modification: Attn = Attention; LN = LayerNorm; Act = Activation; Train = Training/fine-tuning.

Table 7: Coverage of Models and Datasets Used for Accuracy Evaluation of Private Transformer Frameworks

	Framework	Ref	Language Model							Vis. Model		Language Dataset										Vision Dataset			
			BERT-tiny	BERT-base	BERT-large	GPT-2-base	LLaMA2-7B	LLaMA3-8B	Others	ViT-base	Others	SST-2	QNLI	MRPC	RTE	CoLA	MNLI	STS-B	QQP	Wikitext-103	LAMBADA	Others	CIFAR-10	CIFAR-100	Tiny-ImageNet	Others
2PC Interactive	THE-X	[67]	●	○	○	○	○	○	○	○	○	●	●	●	●	○	●	●	●	○	○	○	○	○	○	○
	MPCViT	[82]	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	●	●	●	○
	Iron	[28]	●	○	○	○	○	○	○	○	○	●	●	●	○	○	●	○	○	○	○	○	○	○	○	○
	Primer	[77]	○	●	○	○	○	○	○	○	○	●	○	●	○	○	●	○	○	○	○	●	○	○	○	○
	Liu–Liu	[74]	●	○	○	○	○	○	○	○	○	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○
	CipherGPT	[12]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	East	[59]	●	●	●	○	○	○	○	○	○	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○
	PriViT	[85]	○	○	○	○	○	○	○	●	●	○	○	○	○	○	○	○	○	○	○	○	●	●	●	○
	BumbleBee	[29]	○	●	○	○	○	○	○	●	○	○	●	○	●	●	○	○	○	○	○	○	○	○	○	●
	BOLT	[31]	○	●	○	○	○	○	○	○	○	●	○	●	●	○	○	●	○	○	○	○	○	○	○	○
	CipherFormer	[89]	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○
	PPML-Quantized	[95]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	SecureGPT	[56]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Nimbus	[39]	○	●	○	○	○	○	○	○	○	●	●	●	●	●	●	●	●	○	○	○	○	○	○	○
	PrivCirNet	[37]	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	●	●	●	○
	FastLMPI	[72]	○	●	○	○	○	○	○	○	○	●	○	●	●	○	○	○	○	○	○	○	○	○	○	○
	MPCache	[100]	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○
	CipherPrune	[98]	○	●	○	○	○	○	○	○	○	●	●	●	○	●	○	○	○	○	○	○	○	○	○	○
	Post	[106]	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○
	BLB	[42]	○	●	●	●	○	○	○	○	○	●	●	○	●	○	○	○	○	○	○	○	○	○	○	○
	PrivTF	[44]	○	●	○	○	○	○	○	○	○	●	●	●	○	○	●	○	○	○	○	○	○	○	○	○
	CryptoMoE	[111]	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○
	SecMoE	[112]	○	○	○	○	○	○	●	○	○	○	●	○	●	●	○	○	○	○	○	○	○	○	○	○
	PCFormer	[86]	○	●	○	○	○	○	○	○	○	●	●	●	○	○	●	○	○	○	○	○	○	○	○	○
	SecViT	[92]	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	●	●	●	○
	SMASH	[122]	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○
2PC Noninteractive	PolyTransformer	[87]	○	○	○	○	○	○	●	●	●	○	○	○	○	○	○	○	○	●	○	●	●	○	●	○
	NEXUS	[13]	○	●	○	○	○	●	○	○	○	●	●	○	●	○	○	○	○	○	○	○	○	○	○	○
	Rovida–Leporati	[66]	●	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	Powerformer	[34]	○	●	○	○	○	○	○	○	○	●	○	●	●	○	○	○	○	○	○	○	○	○	○	○
	Rho *et al.*	[65]	○	○	○	○	○	○	●	○	○	●	●	●	●	●	○	●	○	○	○	○	○	○	○	○
	THOR	[33]	○	●	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○
	EncryptedLLM	[48]	○	○	○	●	○	○	○	○	○	●	○	○	○	○	●	○	○	○	○	●	○	○	○	○
	MOAI	[36]	○	●	○	○	○	○	○	○	○	●	●	○	●	○	○	○	○	○	○	○	○	○	○	○
	ENSI	[64]	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○
	Cachemir	[99]	○	○	○	○	○	●	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○
	Euston	[40]	○	●	○	○	○	●	○	○	○	●	●	○	●	○	○	○	○	○	○	○	○	○	○	○
(2+1)PC	MPCFormer	[69]	○	●	○	○	○	○	○	○	○	●	●	●	●	●	●	●	●	○	○	○	○	○	○	○
	RNA-ViT	[70]	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	●	●	●	○
	MERGE	[104]	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○
	SIGMA	[51]	●	●	●	●	●	○	●	○	○	●	●	●	○	○	○	○	○	○	●	○	○	○	○	○
	SAL-ViT	[71]	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	●	●	●	○
	SecFormer	[73]	○	●	●	○	○	○	○	○	○	○	●	●	●	●	○	●	○	○	○	○	○	○	○	○
	PermLLM	[79]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Marill	[93]	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○
	SHAFT	[47]	○	●	○	○	○	○	○	○	○	●	●	○	○	●	○	○	○	○	○	○	○	○	○	○
	Fission	[80]	●	●	●	●	○	○	○	○	○	●	●	○	○	○	○	○	○	○	●	○	○	○	○	○
	SHARK	[121]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Comet	[115]	○	○	○	○	●	○	●	○	○	●	●	●	●	●	●	●	●	○	●	●	○	○	○	○
	CryptPEFT	[83]	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	●	●	○	○
	SOAL	[60]	○	○	○	●	●	○	●	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○
	FuseFSS	[62]	●	●	●	●	○	○	●	○	○	●	●	●	○	●	○	○	○	○	●	○	○	○	○	○
3PC	PUMA	[46]	○	●	●	●	○	○	●	○	○	○	●	○	●	●	○	○	○	●	○	○	○	○	○	○
	Privformer	[88]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Ditto	[75]	○	●	●	●	○	○	○	○	○	○	●	○	●	●	○	○	●	●	○	○	○	○	○	○
	Lu *et al.*	[96]	○	●	○	○	○	○	○	○	○	●	●	●	●	○	●	●	●	○	○	○	○	○	○	○
	Mosformer	[52]	○	●	○	●	○	○	○	○	○	○	●	○	●	○	○	●	○	●	○	○	○	○	○	○
	PrivMLLM	[116]	○	○	○	○	○	○	●	○	●	○	○	○	○	○	○	○	○	○	○	●	○	○	○	●

All "-" = Accuracy is not reported. The table records evaluation coverage, not directly comparable accuracy or performance across frameworks. The first eight datasets in "Language Dataset" belong to the standard GLUE benchmark [117].