SoK: Confidential Transformer Inference and Retrieval-Augmented Generation

Mechanisms, Threat Models, and the Performance Frontier

Timofey Yaluhin
ChainSafe Systems
timofey@chainsafe.io

July 28, 2026

Abstract

Running Transformer inference and retrieval-augmented generation (RAG) over confidential data forces a choice: either expose prompts and documents to a cloud operator, or keep the data on-premises, which confines the deployment to weaker self-hosted models. Existing defenses span five mechanism families: secure computation (MPC and FHE), trusted execution environments (TEEs), static obfuscation, differential privacy, and hybrid TEE-and-obfuscation splits. No prior systematization compares them on a common footing of mechanism, threat model, and deployment cost, and none covers the RAG retrieval layer. We organize the field by *deployment readiness*: the likelihood a scheme is adopted in practice, scored on performance, utility, and threat-model fit. The scoring spans inference and RAG retrieval, both dense and graph. We find that no family dominates: each attains at most two of the three criteria, and which one it sacrifices is fixed by its security basis, so the deployable choice is set by the constraint an application can least afford to relax. Even trusted hardware is no exception, since every surveyed scheme ignores the side channels to which it is most exposed. We further surface hidden deployment costs, such as client reliance and a custom serving path, identify private graph-RAG as the least-served setting, and find that no design yet keeps a pipeline confidential from query to answer.

1 Introduction

1.1 Motivation

Organizations in regulated domains increasingly want to run Transformer inference, both embedding and generation, over confidential data: clinical notes, financial records, legal files, proprietary code. The default path exposes exactly that data, because a hosted inference or embedding API receives every document and query in plaintext, in reach of the operator, a privileged insider, or anyone who breaches the service. GDPR, HIPAA, and the EU AI Act make this incompatible with processing personal, health, or privileged data, and data-residency rules can forbid the egress outright. In a 2024 industry survey, half of enterprises already adopted RAG to keep data in house [14], accepting weaker self-hosted models because the strongest models are reachable only through such APIs.

Confidential inference aims to lift that constraint: to use an untrusted operator's infrastructure, including frontier models and accelerators, without revealing the data to it. The difficulty is protecting data *in use*, not just at rest. On hardware it controls, the operator sees the inputs and activations of every forward pass, and a RAG pipeline further exposes the query, the access pattern over the corpus, and the assembled prompt, where the most sensitive context is concentrated. The European Data Protection Supervisor warns that RAG systems can disclose personal data held in their corpus [19], and a single poisoned document has been shown to turn a RAG application into a remote data-exfiltration channel [24].

A fast-growing body of work answers this with cryptography, trusted hardware, obfuscation, differential privacy, and hybrids, spanning the inference stages (embedding, reranking, generation) and the retrieval and storage layer. But the schemes assume different adversaries and report costs on different hardware, models, and metrics, so a practitioner cannot tell which is deployable. This systematization of knowledge (SoK) supplies that comparison.

1.2 Scope and Positioning

Our organizing goal is *deployment readiness*: how likely a scheme is to be adopted in a real deployment, on three criteria: **performance** (overhead and latency), **utility** (impact on output quality), and **threat-model fit** (an adversary model *appropriate* to the deployment, not maximal). The aim is not to crown a winning family but to make this trade explicit; which schemes sit near the deployable frontier is a *finding* of the synthesis (§9), not an assumption. Threat-model fit is set by the taxonomy of § 2.1; performance and utility are reported per family

Table 1: This SoK against prior surveys and SoKs on related ground. • = covered, ○ = partial/shallow, "—" = absent. *Threat*: threat-model treatment. *Cost*: quantitative cross-scheme performance axis. *Deploy.*: deployment-readiness assessment (• = quantitative scoring, ○ = qualitative discussion). *Code*: code/reproducibility (• = re-benchmark audit, ○ = notes availability). *Infer.*, *RAG* (retrieval/storage), *Graph* (graph-RAG): coverage of each layer.

Survey / SoK	Mechanism families	Threat	Cost	Deploy.	Code	Infer.	RAG	Graph
Bodea et al. [7]	RAG risks (attacks)	risk-taxon.	—	—	—	—	○	—
Li et al. [39]	MPC/FHE	semi-honest	•	—	—	•	—	—
Luo et al. [45]	MPC/FHE (+training)	—	—	—	○	•	—	—
Xue et al. [75]	FHE only	—	•	—	—	○	—	—
Ma et al. [47]	DP/obfusc. (hi-lvl)	lifecycle	—	—	—	○	—	—
Andreoletti et al. [4]	crypto+TEE+info-red. (fam.)	operational	○	○	—	•	—	—
Chen et al. [11]	MPC/FHE/hybrid	semi-honest	•	○	•	•	—	—
This SoK	**all five**	5-axis	•	•	○	•	•	•

(§3–§7) and drawn together in §9.

No prior survey occupies this ground (Table 1); each covers a slice. A risk-centric SoK catalogs attacks over naive RAG [7]; a cluster of surveys treats *cryptographic* private inference, confined to MPC/FHE [11, 39, 45]; others go deep on a single FHE primitive [1, 75] or stay high-level across the lifecycle [47]. For MPC/FHE-inference protocol depth we *refer the reader to* Li et al. [39] and Chen et al. [11] (the latter adding a reproducibility re-benchmark), keeping only a few frontier exemplars (§3). The one prior work sharing our deployability lens, Andreoletti et al. [4], differs substantively: it is a *meta*-survey at the level of privacy-enhancing-technology (PET) *families* (qualitative trade-offs, thin on information-theoretic and obfuscation methods, no retrieval or graph-RAG), and where it argues one trust-minimizing trajectory (TEE → FHE) we score *individual schemes* and reach a different conclusion: today's deployable options are present-day systems such as the hybrid split, obfuscation, and confidential-accelerator TEEs, not intermediate stages toward a cryptographic endpoint (§ 9.2). Our contribution is thus breadth across all five families, per-scheme depth, the retrieval and graph-RAG dimension, and a quantitative deployment-readiness lens.

Two structural choices follow. First, a *hardware-trust* ladder, defined with the trust-anchor axis of our taxonomy (§ 2.1), feeds the performance and threat-fit criteria: Tier 1 (no trusted hardware), Tier 2 (CPU-TEE only, commodity GPU untrusted), and Tier 3 (a confidential accelerator trusted end-to-end). We focus on Tiers 1 and 2, where confidentiality is an open *design* problem, and keep **Tier 3** as a performance baseline (§4): once a confidential accelerator is trusted end-to-end, confidentiality largely reduces to hardware availability, enclave construction, and attestation, a narrower, hardware-specific space systematized in its own right [69]. Second, the **practical-frontier** restriction follows directly from the deployment lens: comparison tables hold only state-of-the-art schemes with deployable overhead, and foundational or superseded work is cited for lineage only.

The systematization spans Transformer inference (embedding, reranking, generation) and RAG including *graph-RAG*, a setting the prior RAG-privacy SoK [7] excludes. We treat a semi-honest cloud as the baseline adversary, documenting the range to malicious in § 2.1. Explicitly *out of scope*: microarchitectural side-channels (a field-wide exclusion we make a finding of, § 2.1), full verifiable computation (lightweight composable integrity is reported when present, but zkLLM-class proving is not covered), and foundational/superseded schemes.

1.3 Contributions

This paper is a *systematization of knowledge*. Concretely:

- **A five-axis threat-model taxonomy** (§ 2.1) that classifies confidential Transformer-inference and RAG schemes by adversary behavior, trust anchor, attacker knowledge, confidentiality scope, and *security basis* (cryptographic / DP / hardware / heuristic), the last separating provably-secure schemes from empirically-resistant ones, and surfacing the field-wide side-channel blind spot we report as a finding in § 9.3.
- **A mechanism-organized systematization** of five inference-defense families (§3–§7) plus a dedicated treatment of the confidential **retrieval and storage** layer (§8), dense and graph/LightRAG, spanning the full inference-and-RAG pipeline.
- **A performance comparison** synthesized from reported numbers (§9), the cost axis the existing RAG-privacy SoK [7] omits, with an explicit non-comparability caveat.
- **A deployment-readiness synthesis** (§9) that scores each family on performance, utility, and threat-model fit, showing that real-world readiness comes from balancing the three rather than maximizing any one, and locating each family on that frontier.
- **A stage × mechanism gap-map** (§9) whose empty cells are the field's open problems.

1.4 Organization

The evaluation framework (§2) defines the security taxonomy and the cost metrics. The five mechanism families for

confidential *inference* are systematized in §3–§7; the RAG *retrieval and storage* layer (dense and graph/LightRAG, across all mechanisms) is treated separately in §8, since its trade-offs and primitives differ. The synthesis (§9) draws them together: the stage × mechanism gap-map, the performance comparison, the deployment-readiness profile, graph-RAG as a retrieval setting, and the field-wide side-channel finding.

2 Evaluation Framework: Security and Cost

We compare confidential Transformer-inference and RAG schemes along two kinds of dimension. *Security* captures categorical properties of who the adversary is and what stays hidden (§ 2.1); *cost* is the quantitative price of that protection, in performance and utility (§ 2.2). This single split runs through the paper: the security dimensions yield the threat-model-fit criterion of our deployment-readiness lens (§1). We also fix here how schemes are *grouped* for comparison: by use case, not by family (detailed in § 2.2).

2.1 Taxonomy: Threat Model and Security

Confidential Transformer-inference and RAG systems are not built against a single adversary. Each scheme makes its own assumptions about who the attacker is, what it knows, what it can observe, and what must stay hidden. Those assumptions, not the mechanisms alone, decide whether a defense is adequate. We classify the security of each scheme along five *categorical* axes (§ 2.1.1), place every inference scheme on them (§ 2.1.2), and treat the attacker's *capability budget* as a cross-cutting concern (§ 2.1.3). A field-wide side-channel blind spot emerges from the exercise (§ 9.3).

2.1.1 Axes

We classify each scheme along five axes.

1. Adversary behavior. *Honest-but-curious* (HbC): the operator runs the protocol faithfully but inspects every observable for private information, the standard cloud model, adopted by most schemes here. *Malicious/active*: the operator may additionally deviate (drop, replay, reorder, tamper), requiring integrity machinery on top of confidentiality. Behavior decides whether confidentiality alone suffices.

2. Trust anchor (TCB). What the scheme roots its guarantees in: *none* (purely cryptographic, statistical, or obfuscation-based); a *CPU-TEE* (Intel TDX, AMD SEV-SNP); a *GPU-TEE* (a confidential H100/H200/B200-class accelerator); or *multi-party non-collusion* (the MPC assumption). The trusted computing base (TCB) is the set of components that must be trusted not to leak; a larger TCB means more hardware is trusted, so a purely cryptographic scheme has the smallest TCB and a GPU-TEE among the largest. A larger TCB trades a stronger trust assumption for performance or simplicity. These anchors collapse onto a three-level *hardware-trust* ladder we use to organize the systematization and its scope (§1):

Tier 1. *No trusted hardware*: confidentiality rests on cryptography or statistics alone (MPC/FHE, static obfuscation, DP). "None" constrains what must be *trusted not to leak*, not where the *work* runs. Several Tier-1 schemes are not thin-client: client-side private information retrieval or oblivious RAM (PIR/ORAM) retrieval, or two-party computation (2PC) where the user is a compute party, make the client load-bearing, a deployment factor we revisit in §9.

Tier 2. *CPU-TEE only*: a commodity CPU enclave is trusted; any GPU is untrusted.

Tier 3. *Confidential accelerator*: a confidential GPU or accelerator-TEE trusted end-to-end.

The Tier-2→Tier-3 jump (trusting the accelerator silicon end-to-end) is the sharpest escalation on this axis, so a scheme's tier is the single most consequential entry in its row; the hybrid split's defining move (§7) is to deliver Tier-3-class GPU performance at Tier 2.

3. Attacker knowledge. *Black-box* (the adversary sees only the input/output interface) or *white-box* (it can additionally inspect the model weights and internal activations, the realistic case for open-weight Transformers). We adopt Kerckhoffs's principle throughout: in either regime the adversary knows the algorithm and is denied only a fresh per-session secret, whether an obfuscation transform, injected DP noise, or cryptographic keys. That secret therefore belongs to the *security basis* (axis 5), not to a separate knowledge level. White-box is the regime in which static schemes are most stressed: an obfuscation transform secure only while the weights are private collapses once an attacker who knows the public weights can solve for it (§5).

4. Confidentiality scope. What the scheme keeps hidden and what it leaves exposed are complementary. The protected *asset* may be the input, the input *and* output, the model weights, or the access pattern over a corpus. The complement is the *residual leakage*: what the adversary still observes despite the scheme (obfuscated I/O, mixed activations in untrusted VRAM, bounded access patterns, or distance structure). Microarchitectural, cache, timing, and power side channels are a residual category *every* surveyed scheme places out of scope, a uniformity we elevate to a finding (§ 9.3) rather than carry as a per-scheme cell. Two schemes agreeing on every other axis can diverge sharply here, and it is the residual

Table 2: Threat models for the inference schemes of Table 3, on the five axes of § 2.1.1; rows mirror that table's families. *Knows* (attack surface): `wb`=white-box (model weights and activations inspectable); `bb`=black-box (input/output interface only). *Protects* and *Residual leak* are the two facets of confidentiality scope (asset hidden from the operator vs. what it still observes): `ap`=access pattern, `act`=activations in VRAM, `ct`=ciphertext. *Basis*: crypto=cryptographic, DP=differential privacy, HW=hardware-enforced, heur.=heuristic. Side channels are out of scope for every row (§ 9.3).

Scheme	Behavior	Trust anchor	Knows	Protects	Residual leak	Basis
Cryptographic (MPC/FHE) — §3						
SHAFT	HbC	2PC	`wb`	in+model	shares	crypto
Fission	HbC	MPC evaluators	`wb`	in+model	shuffled chunks	crypto+heur.
Euston	HbC	none (FHE)	`wb`	in+out	`ct` only	crypto
PermLLM	HbC	2PC+dealer	`wb`	in+model	shares	crypto+heur.
CryptoMoE	HbC	2PC	`wb`	in+model	shares	crypto
TEE — §4						
Nvidia-CC	malicious	GPU-TEE	`bb`	data+model	`ap`	HW
Ascend-CC	malicious	NPU-TEE	`bb`	data+model	`ap`	HW
PipeLLM	HbC	GPU-TEE	`bb`	data+model	PCIe `act`	HW
Opal	malicious	CPU+GPU-TEE	`bb`	data+access	op-type	crypto+HW
Static obfuscation — §5						
AloePri	HbC	none	`wb`	in+out	obf. I/O+weights	heur.
SGT	HbC	none	`wb`	input	obf. embed.	heur.
OSNIP	HbC	none	`wb`	input	obf. embed.	heur.
Eguard	HbC	none	`wb`	input	obf. embed.	heur.
KV-Cloak	HbC	none	`wb`	input	obf. KV-cache	heur.
ObfusLM	HbC	none	`wb`	in+out	anon. embed.	heur.
CodeCipher	HbC	none	`wb`	input	obf. tokens	heur.
Differential privacy — §6						
DP-Forward	HbC	none	`wb`	input	noised embed.	DP
SPARSE	HbC	none	`wb`	input	noised embed.	DP
NVDP	HbC	none	`wb`	input	noised embed.	DP
DP-KSA	HbC	none	`bb`	corpus	DP answers	DP
DP-Fusion	HbC	none	`bb`	context	DP output	DP
Hybrid split (TEE+Obfuscation) — §7						
GELO	HbC	GPU-TEE	`wb`	input	`act`	heur.+HW
TwinShield	HbC+verif.	CPU-TEE+SS	`wb`	data+model	`act`	crypto+HW
ObfuscaTune	HbC	CPU-TEE	`wb`	data+model	`act`	heur.+HW
SCX	HbC	CPU-TEE	`wb`	input	obf. KV-cache	heur.+HW
CMIF	HbC	CPU-TEE	`wb`	input	DP tokens	DP+HW

leakage, not the stated protection goal, that an attack ultimately exploits.

5. Security basis. What the guarantee *rests on*, independent of *which* entity is trusted (axis 2): *cryptographic* (provable under a hardness assumption, as in FHE/MPC under LWE/OT); *information-theoretic* (no computational assumption, as in one-time masking); *differential privacy* (a formal (ε, δ) or metric bound, where the privacy budget ε caps how much any single record can shift the output and δ is a small failure probability); *hardware-enforced* (reduces to TEE isolation and attestation); and *heuristic* (resistance shown only experimentally, as in most static obfuscation). This axis separates schemes the others cannot: an FHE scheme and a static-obfuscation scheme can share an identical row on axes 1–4 yet differ as cryptographic indistinguishability versus an empirically measured resistance that can be far weaker than reported. Security basis is thus the sharpest predictor of whether a confidentiality claim survives scrutiny, and the axis that *threat-model fit* (§1) weights most heavily.

2.1.2 The Landscape

Table 2 places every inference scheme of Table 3 on the five axes. Four patterns stand out. First, the obfuscation family clusters tightly at *HbC + white-box*: these schemes explicitly assume the attacker knows the open-source model and the method, and depend entirely on an ephemeral secret. Second, the trust anchor varies more than the adversary behavior does: schemes that face nearly the same attacker still root trust in very different places (nothing, a CPU-TEE, or a GPU-TEE), which is the real axis of disagreement in the field. Third, defending a *malicious* operator tracks the trust anchor: the confidential-accelerator designs (Nvidia-CC, Opal, Ascend-CC) withstand a host that may tamper with computation outside the enclave, and TwinShield an accelerator that may deviate from the protocol (caught by its U-Verify integrity checks); the cryptographic, obfuscation, and DP schemes assume an honest-but-curious operator. Fourth, and most consequential, the *security basis* separates the table where no other axis can: the obfuscation rows rest on heuristic, experimentally-measured resistance, while the cryptographic schemes rest on hardness assumptions and the TEE designs on hardware. Several heuristic and cryptographic schemes alike anchor trust in *nothing* (trust anchor *none*), so this distinction is invisible without the basis axis.

Table 3: **Inference performance across mechanism families**. Cryptographic rows give *absolute* latency (per generated token, "/tok", for decoders; per 128-token forward pass, "/inf", for encoders) with no vs-plaintext multiplier; TEE, obfuscation, DP, and hybrid rows give ×*-overhead* vs. plaintext on the same hardware (communication local/negligible, omitted). An inline "offl." figure marks one-time offline cost. Figures are paper-reported; cross-row figures are indicative, not co-measured (§ 2.2). Utility is the accuracy or recall delta vs. plaintext; obfuscation/DP privacy is treated in §5, §6.

Scheme	Technique	Model	Latency / overhead	Utility
Cryptographic (MPC/FHE) — §3				
SHAFT [31]	2PC	BERT-base (110M)	41/281 s LAN/WAN; 10.5 GB/inf	−0.5%
Fission [66]	MPC+evaluators	ModernBERT (149M)	<5 s/inf; <3 GB/inf	−0.3%
Euston [23]	FHE (non-int.)	BERT-base (110M)	46 s/inf (GPU)	−0.1%
PermLLM [81]	2PC+dealer	ChatGLM-6B	3 s/tok; 20 Mb/tok	exact
CryptoMoE [84]	2PC (MoE)	OLMoE-7B-A1B	22 s/tok; 2.8 GB/tok	−0.8%
TEE — §4				
Nvidia-CC [85]	GPU-TEE (H100/H200)	Llama2 7–70B	1.04–1.08×	exact
Ascend-CC [15]	NPU-TEE (Huawei Ascend)	Llama2/3 7–13B	≈1× (<0.1% @≥2K seq)	exact
PipeLLM [59]	GPU-TEE+CVM	OPT 13–175B	≈1.2× (<19.6%)	exact
Opal [30]	GPU-TEE + Ring-ORAM	KG+ANN retr. (B200)	1.57× (2.32 s/q)	+13% vs ANN-only
Static obfuscation — §5				
AloePri [43]	covariant data+weight	Qwen/Llama ≤671B	≈1× (offl. transform, mins)	−0 to −3.5%
SGT [54]	learned embed. transform	Llama-1B/Qwen3	≈1×; offl. train (hrs–days)	−0.4%
OSNIP [8]	semantic null-space	Llama-3.2/Qwen3	≈1× (0.96 ms); offl. train	≈0
Eguard [44]	MI-loss projector	T5/MPNet/RoBERTa	≈1.7×; 1.6–3.4× train step	−0 to −2%
KV-Cloak [46]	KV-cache per-block perm	LLaMA-3.1-8B	≈1× (<0.45% prefill)	exact
ObfusLM [42]	(k, ε)-anon embed+head	BERT/Llama3-8B	≈1× (1.29 s/inf); >10 h FT offl.	−4% classif. / −6% gen.
CodeCipher [40]	token confusion map (code)	CodeLlama-7B	≈1× (offline opt.)	−3% Pass@1
Differential privacy — §6				
DP-Forward [17]	matrix-Gaussian fwd	BERT enc.	≈1× (offl. FT)	92% SST-2 @ε≈16
SPARSE [65]	metric-DP mask	GTR/SBERT enc.	≈1× (offl. train 25 min/10k)	65% STS-12 util @ε=10
NVDP [18]	learned VIB (Bayes-DP)	BERT-base (GLUE)	≈1× (offl. NVIB)	83% MRPC @ε_μ≈11
DP-KSA [60]	output-DP (sample-aggr.)	Qwen2.5-3B/Llama	≈N× gen. (N≈80)	F1 +3.5 vs no-RAG @ε=8
DP-Fusion [62]	output-DP (dist. blend)	Qwen2.5-7B	m+1 pass/tok (≈1× lat.)	ppl 1.43 vs 1.03 @ε≈16
Hybrid split (TEE+Obfuscation) — §7				
GELO [5]	GPU-TEE + per-batch mix	Llama-2-7B	1.2–1.3× (per offload)	exact
TwinShield [74]	CPU-TEE + SS + U-Verify	BERT/ViT/LLaMA	−4–6× vs CPU-TEE	exact
ObfuscaTune [21]	≈5% params in TEE	GPT-2	1.5–4.3×	−1%
SCX [80]	per-session OTK KV-cache	7B decoder	≈1×	exact
CMIF [78]	CPU-TEE embed + RNM-DP	Llama2/3 7–8B	1.3–1.4×	84% SST-2 @ε=3

2.1.3 The Attacker's Capability Budget

The five axes say what an adversary *is*, not how *much* it can spend. Yet the capability budget (compute, and especially the observation horizon: a single inference vs. many adaptive runs) is decisive, because the surveyed schemes target markedly different budgets. AloePri is explicit that it "aims to provide adjustable security for constrained attackers in real-world scenarios, rather than offering ideal-world security guarantees against worst-case attackers" [43]; GELO's guarantee is per-batch non-identifiability under a fresh mixing matrix, conceding nothing *across* batches but leaving a single-batch blind-source-separation problem [5]. A worst-case adversary with unbounded compute and unlimited adaptive observation defeats every static-obfuscation scheme here; an evaluation assuming one would measure nothing interesting and misrepresent schemes that never claimed it. This is why *threat-model fit*, not maximality, is a deployment-readiness criterion: a scheme defending an adversary stronger than its deployment faces overpays in performance, one defending a weaker adversary fails to protect, and the deployable point is the calibrated one, on which we score families in §9.

We therefore treat the budget as an explicit per-scheme parameter, assessing each scheme *within its own constrained-attacker regime* (white-box on weights and method, denied the per-session secret, under the compute and observation horizon the scheme's own evaluation permits), so that a reported break is a break by the victim's own standard, not an artifact of an inflated adversary.

2.2 Cost: Performance and Utility

The other two deployment-readiness criteria are quantitative: *performance* (overhead and latency) and *utility* (impact on output quality). Comparing them across papers is hard to do fairly: reported numbers come from different hardware, model sizes, datasets, and metrics, so a raw side-by-side can mislead. We fix here a common set of *reference metrics* for both, and state how we group schemes when we compare them.

Normalizing performance. Wherever a paper gives a plaintext baseline we express cost as an *overhead multiplier* (× vs. plaintext on the same hardware and model); otherwise we compare only *within a regime*, a (model-size × hardware) bucket (e.g. BERT-base on one GPU; a 7B decoder over WAN), never across regimes. Cross-scheme figures are *indicative* by default (each from its own testbed: an order of magnitude, not a precise ratio); only figures

we flag *co-measured* (benchmarked together in one paper) support direct comparison. Even within a regime, a comparison is fair only when the number of parties, the numerical precision, and the network setting match. Chen et al. [11] find WAN-tuned schemes miss their reported numbers under other network profiles, so LAN and WAN figures are not interchangeable.

Normalizing utility. Schemes that compute the true function (cryptography, TEE, invertible-mask hybrid splits) are *exact*; approximate schemes (obfuscation, DP, polynomial-approximation MPC) are reported as a *delta* against the plaintext baseline in the use-case-appropriate metric, so a retrieval recall drop and a generation accuracy drop are never conflated.

Grouping: by use case, not by family. We *systematize* by mechanism family (§3–§7) but *compare* performance and utility **by use case**, since a deployment decision weighs every mechanism that serves a stage against the others (private generation pits MPC, TEE, obfuscation, and hybrid split against each other, not FHE against obfuscation in the abstract). We use four use cases aligned with the gap-map stages (§9): *generation* (decoder) and *embedding* (encoder) inference, and retrieval split by index structure into *vector* (flat/IVF ANN and vector storage) and *graph* (graph-based ANN and graph-semantic search), which impose different costs and threat surfaces; each scheme is grouped by the model class it evaluates. *Reranking* is cross-encoder inference, so it inherits the embedding and generation schemes (only Opal exercises it directly, in its enclave pipeline); no surveyed work optimizes a standalone private reranker, which we leave as an open problem. We give it no separate cost bucket, and the gap-map (§9) shows it covered by the inference families. Each use case has a canonical metric: latency and ×-overhead for generation and embedding; latency/query and communication at a fixed corpus size for retrieval; and utility as the task-accuracy (generation/embedding) or recall@k (retrieval) delta against the plaintext baseline.

3 Cryptographic Inference: MPC and FHE

Cryptographic schemes compute on data the server never sees in the clear: secure multi-party computation (MPC) and secret sharing (SS) split the computation across non-colluding parties, while fully-homomorphic encryption (FHE) evaluates directly on ciphertext. They offer the strongest confidentiality in our taxonomy (§ 2.1): a trusted computing base of *none* (FHE) or *non-colluding parties* (MPC/SS), with no special hardware. That guarantee costs one-to-three orders of magnitude of overhead in latency or communication, concentrated in the non-linear operations. Because MPC/FHE *inference* protocols are already covered by dedicated surveys, we do not re-systematize them: Li et al. [39] give a layer-wise MPC-vs-FHE treatment and Chen et al. [11] a reproducibility re-benchmark. We cover the sub-family briefly through five frontier exemplars (Table 3) that locate it on our deployment-cost frontier, and defer the cryptographic *retrieval* layer (PIR/ORAM/SSE/DCPE) to §8. On our criteria, cryptographic inference scores highest on threat-model fit and utility but lowest on performance (§9).

3.1 Approaches

Secret sharing and MPC split each value into shares held by non-colluding parties. Linear maps (the bulk of a Transformer's matrix multiplications) are local or need only cheap Beaver-triple multiplications, so cost concentrates in the non-linear operations (Softmax, GELU, LayerNorm), which must be evaluated through oblivious transfer or garbled circuits, or else replaced by polynomial approximations. Each of these options adds rounds, communication, or error. *FHE* instead evaluates directly on ciphertext under one key; the RNS-CKKS scheme used in most FHE Transformer inference is *leveled*, bounding multiplicative depth before an expensive bootstrap and again forcing the non-linearities into low-degree polynomial approximations. In both families these non-linear operations dominate: Li et al. [39] measure them at >85% of MPC runtime and ≈60% for FHE. Autoregressive decoding compounds the cost, since each generated token is a full forward pass whose per-pass overhead and interaction rounds are multiplied by the generation length [4].

Our five exemplars (Table 3) span the design space: 2PC (SHAFT), 2PC with an offline dealer (PermLLM), distributed MPC with evaluator nodes (Fission), private mixture-of-experts routing (CryptoMoE), and non-interactive FHE (Euston). CryptoMoE is the one architecture-specific case: it keeps MoE expert-routing secret-shared. This hides a routing pattern that leaks the input's semantic type and that has been used to recover a co-batched prompt in full [77]. Exploiting that sparsity, CryptoMoE realizes a 2.8–3.5× end-to-end latency reduction (and 2.9–4.3× in communication) over a dense-MPC baseline. Across all five, even the fastest exchanges hundreds of MB to GB per token or inference, with output that is exact (MPC) or near-exact (FHE). No scheme's source paper reports its overhead as a slowdown factor against a plaintext baseline, so Table 3's figures are the papers' reported absolute costs or 1-hop extrapolations (§ 2.2).

3.2 Limitations and Known Attacks

Each failure mode below is a concrete attack on a specific trust assumption or efficiency heuristic; Table 4 collects them by family (MPC/secret sharing vs. FHE).

Table 4: Known attacks on cryptographic *inference*, separated by family (MPC/secret sharing vs. FHE; the failure modes differ). Each row names the attack, the assumption or efficiency heuristic it exploits, the schemes at risk, and the defense. IND-CPA$^{\mathrm{D}}$ = indistinguishability under chosen plaintext with a decryption oracle.

Attack (cited)	Assumption / heuristic exploited	Schemes at risk	Defense
MPC / secret sharing			
Model extraction (Muse) [35]	malicious client deviating from a semi-honest 2PC protocol	semi-honest 2PC inference (SHAFT, PermLLM)	malicious-client security: SIMC [10] (1.3–2×)
Party collusion	two parties collude ⇒ additive shares recombine to plaintext	MPC/SS (PermLLM, Fission)	non-collusion deployment
Permutation inversion (Fission) [66]	heuristic, non-information-theoretic permuted-plaintext reveal	PermLLM-style permutation	multi-evaluator chunking ($\sim \log N$) + query batching
Prompt reconstruction (Hidden No More) [63]	fixed permutation + decoder injectivity	STIP, Centaur, PermLLM	token sharding [64] (statistical only)
FHE			
Approx.-ciphertext key recovery (IND-CPA$^{\mathrm{D}}$) [36]	adversary observes decryptions of approximate CKKS ciphertexts	CKKS w/ exposed decryptions (not non-int. Euston)	noise flooding (precision cost)

The trust assumption is the security. Every scheme here is proved secure only against a *semi-honest* adversary, and MPC/SS additionally require non-collusion. Neither is free: Muse shows a single *malicious* client can extract the server's weights from a semi-honest 2PC protocol [35], and two colluding parties can recombine additive shares to recover the plaintext. Closing the malicious-client gap costs ≈1.3–2× (SIMC [10]) and full malicious security more; no scheme we survey adopts either. So under a realistic cloud adversary that can deviate or covertly co-locate the nominally non-colluding parties, the guarantee degrades to that of an honest operator. This is comparable in trust to a TEE, but without the attestation that lets a client *verify* it (§4, §7).

Permutation secrecy is heuristic, not information-theoretic. The fastest exemplars gain speed by evaluating non-linearities on *permuted* plaintext rather than under MPC (PermLLM reveals permuted hidden states; Fission reveals shuffled activation chunks to evaluators), which both concede is not information-theoretic. Hidden No More reconstructs prompts near-perfectly from *fixed* permutations via decoder injectivity, breaking STIP, Centaur, and PermLLM in a third-party-node deployment [63]; the co-designed token-sharding defence is only *statistically* secure [64]. Exposure is contingent on *who* sees the reveal. PermLLM reveals to the user (their own data), and Fission's chunking reduces leakage as $\sim \log N$ in the number of evaluators N. Permutation is therefore best characterized as a heuristic speed/leakage trade-off, not a guarantee on par with full MPC or FHE.

FHE: depth budget and approximate-ciphertext security. RNS-CKKS's bounded multiplicative depth forces range-sensitive polynomial approximations of the non-linearities. Separately, approximate-FHE ciphertexts carry a subtlety: the IND-CPA$^{\mathrm{D}}$ attack recovers a CKKS secret key when an adversary can *observe decryptions* of honestly evaluated ciphertexts [36], mitigated by noise flooding. It applies only where decrypted outputs are exposed; the non-interactive inference schemes here (the server never decrypts) are unaffected, but RAG deployments that return decrypted scores to a client should apply the countermeasure (§ 8.6).

4 Trusted Execution Environments

A trusted execution environment runs the computation in plaintext inside a hardware-isolated enclave and proves to the client, by remote attestation, that the expected code and model are what executes. It achieves near-plaintext speed and shifts trust to the silicon vendor, but at full scale it demands either a CPU enclave large enough for the model (Tier 2, impractical for large models) or a *confidential accelerator* (Tier 3). We cover two such accelerators: a confidential NVIDIA GPU (an H100/B200-class GPU-TEE) and a confidential NPU (the Huawei Ascend line). These designs are therefore predominantly Tier 3: once the accelerator is trusted end-to-end, confidentiality reduces to enclave provisioning and attestation, a hardware-specific space systematized separately [69]. The corpus's strongest adversary, a *malicious host* that may tamper with computation outside the enclave, is met here under three different trusted bases. NVIDIA's confidential GPU [85] trusts a CPU-TEE confidential VM (CVM; SEV-SNP/TDX) as its control peer, so its guarantee is only as strong as that CPU-TEE and the access pattern still leaks; Opal [30] keeps that base but adds ORAM, hiding the access pattern obliviously rather than under a hardness assumption; Ascend-CC [15] drops the CPU-TEE, leaving the NPU as the sole trusted base against a fully compromised host. Pure TEE attains the highest performance and utility in the survey, but its Tier-3 hardware dependence is the binding deployability constraint; we rank it against the other families in §9.

4.1 Approaches

End-to-end confidentiality is a deployment workflow, not the enclave alone: it binds the hardware guarantee to the threat model [4]. The workflow has four parts: *remote attestation*, where the client verifies the enclave's measurement (the hash of the loaded code, model, and configuration); *key provisioning*, where session keys are released to the enclave only after attestation succeeds; a *secure channel*, where prompts are decrypted and processed only inside the enclave and completions are re-encrypted before leaving it; and *operational discipline*, where plaintext logging, tracing, and telemetry outside the boundary are forbidden by policy rather than enforced by the hardware. In the taxonomy of §2 the trusted core is a CPU-TEE or a confidential accelerator, the adversary ranges from honest-but-curious to the malicious host above (by trusted base), and the residual surface is what escapes the boundary: token-streaming timing and length, and the microarchitectural side channels of § 9.3. The designs differ in which hardware holds the enclave.

The simplest design runs the entire model inside a **CPU enclave** (Intel TDX, AMD SEV-SNP). It needs no special accelerator (Tier 2), but it is bounded by the enclave's memory capacity, which caps the model size that fits, and by the absence of a GPU, which throttles throughput, so it is impractical for large models; it is the floor of the family, not a deployment target.

The Tier-3 designs run on a trusted accelerator. **The confidential GPU** [85] places the whole model in a hardware enclave on an H100/H200/B200-class accelerator, encrypts its HBM, and protects the CPU–GPU PCIe link with AES-GCM; paired with an attested CPU confidential VM at the other end, it delivers whole-model confidentiality at near-plaintext speed. **Ascend-CC** [15] places the enclave on a discrete NPU (a Huawei Ascend 910A) instead. Before any plaintext exists, the NPU revokes the host's DMA mappings, decrypts the data and model on-device, and re-encrypts results before remapping; it also attests each operator binary, so a malicious host cannot substitute one that exfiltrates plaintext. Encrypting the model and operator binaries alongside the input lets one deployment serve the data owner, model owner, and cloud at once, with no change to the PyTorch stack. Its trusted base is an on-chip security module and hardware root of trust that a commodity device lacks.

PipeLLM [59] targets the encrypted PCIe channel, the one place a confidential GPU pays: it speculatively precomputes the AES-GCM keystream so the link encryption overlaps computation instead of serializing behind it. **Opal** [30] confronts that malicious host directly. Encryption alone does not stop it, since encryption hides content but not the host's control of scheduling, memory mapping, and message replay; Opal therefore verifies the integrity and freshness of every result against attested, Merkle-authenticated state, rejecting any tampered or rolled-back execution. It additionally hides access patterns with oblivious RAM over an encrypted disk, running an in-enclave knowledge-graph and ANN retrieval pipeline on a TDX host with a B200 accelerator.

4.2 Performance

Speed is near-plaintext, with only single-digit-percent overhead across the band (Table 3): the confidential GPU adds 4–8% on Llama-2 7–70B, and Ascend-CC adds under 0.1% on Llama-2/3 at realistic sequence lengths. These percentages are each platform's own overhead and are not comparable across hardware classes, because they sit on very different absolute baselines. An H100-class confidential GPU delivers on the order of 1000 TFLOPS of dense FP16 over 3.35 TB/s of HBM3, whereas the Ascend 910A is an earlier 7 nm NPU at roughly 256 TFLOPS FP16 with markedly lower memory bandwidth, so its near-zero *relative* overhead is measured against a lower absolute throughput, not evidence that it runs faster. Within a platform the overhead falls as the model and the sequence grow, because it is a fixed per-request charge (channel encryption and a one-time setup) that does not scale with the token count and so becomes a smaller fraction of a larger computation. The charge is visible only on very small models at very short inputs; on Llama-2/3 it stays below 0.1% across sequence lengths.

The residual cost is not the computation but the trust mechanisms around it, and it has two sources. The first is the encrypted CPU–accelerator channel. On today's deployed Hopper-generation GPUs, PCIe transfers are staged through a software bounce buffer that the CPU encrypts with AES-GCM, and the benchmark literature finds this staging, not the GPU computation, to be the dominant source of the residual overhead [13]. The second source is inside the enclave, where memory protection attaches version-number and message-authentication-code (MAC) metadata to each block and walks a Merkle tree on every off-chip access. Two hardware optimizations attack these in turn. *PipeLLM* pipelines the channel encryption, overlapping the AES-GCM with computation to hold a confidential GPU near 1.2× the unprotected latency [59]. *TensorTEE* attacks the in-enclave metadata: a unified tensor-granularity for the CPU and NPU enclaves keeps version numbers on-chip, replaces per-cacheline MACs with delayed-verification tensor-wise MACs, and moves data between enclaves without re-encryption, which brings a heterogeneous CPU–NPU TEE to within 2.1% of non-secure execution, a 4.0× speedup over prior heterogeneous designs [26]. TensorTEE is evaluated on training workloads in architectural simulation rather than on deployed silicon, but the metadata it removes is incurred by any TEE-protected tensor workload, inference included.

These optimizations are software workarounds for a cost that hardware is beginning to absorb. Inline PCIe TEE-IO, the TEE Device Interface Security Protocol (TDISP) and PCIe Integrity and Data Encryption (IDE) standards adopted as Intel TDX-Connect, AMD SEV-TIO, and

Table 5: Attacks against trusted execution environments, banded by whether the hardware vendor treats the attack as in scope. The first band is closed on the normal platform patch cycle or by correct deployment; the second is a class the vendor threat model deliberately excludes and does not fix. The items are hardware and platforms, not only schemes, and TensorTEE and Ascend-CC inherit the same exclusions. This is the cross-cutting finding of § 9.3.

Attack	Target	Effect	Status / defense
Patched or deployment-addressable: closed on the platform cycle or by correct configuration			
Malicious-interrupt injection (WeSee) [56]	SEV-SNP guest via a malicious hypervisor (#VC)	leaks and corrupts enclave state	AMD firmware mitigation
GPU-memory residue (LeftoverLocals) [57]	leaked GPU local memory across processes	reconstructs responses and activations	vendor-patched
Access-pattern leakage [32]	memory and retrieval access trace	reveals which items a query touches	oblivious RAM (Opal)
Unencrypted-HBM residue	confidential-GPU memory left in the clear on some parts	exposes plaintext outside the encrypted boundary	full-memory encryption
Structural: deliberately outside the vendor threat model, not fixed			
Physical memory-bus interposition (TEE.Fail) [61]	DDR5 bus on SGX/TDX/SEV-SNP and confidential-mode GPUs	extracts keys and forges attestation	none: AMD and Intel declare physical attacks out of scope

NVIDIA Blackwell Trusted I/O, encrypts the link in the PCIe controller itself and removes the software bounce buffer entirely, but only when both the CPU and the accelerator support it. Most deployed systems do not yet provide that support [53]. Opal pays more than the rest of the band, for access-pattern hiding rather than for compute. Hiding which records a query touches has two routes inside an enclave: rescan the whole encrypted store on every request ($O(N)$), or use ORAM ($O(\log N)$). Opal's ORAM costs 1.57× the query latency of the same pipeline with no hiding (plaintext storage, selective fetches) [30]. The output is exact throughout, since a TEE computes the true function in plaintext.

4.3 Limitations and Known Attacks

The attacks here split by whether the hardware vendor treats them as in scope, and that line is the section's main caveat (Table 5). Most are actively engineered against and already mitigated. WeSee's malicious-interrupt injection into AMD SEV-SNP is closed by firmware mitigations [56], and LeftoverLocals, which reconstructed responses from leaked GPU local memory, was patched by the affected vendors within months of disclosure [57]. These are the normal patch cycle of a maturing platform, not standing holes, and they apply to the hardware itself, so Ascend-CC inherits the same exposure and the same fixes.

One class, though, the vendors decline to close. Server-grade TEEs dropped the Merkle-tree integrity and replay protection of client SGX in favor of deterministic AES-XTS memory encryption, a deliberate trade-off to support large confidential VMs at low latency. TEE.Fail exploits exactly that determinism: a sub-$1000 DDR5 bus interposer reads and replays memory traffic to extract keys and forge attestation across Intel SGX/TDX, AMD SEV-SNP, and confidential-mode GPUs, and both AMD and Intel classify such physical attacks as out of scope with no mitigation planned [61]. This is a deliberate exclusion, not an oversight, and the long-term cost of anchoring trust in hardware. A guarantee that reduces to "the silicon is correct" holds only within the vendor's threat model, and only until the next attack on the microarchitecture or the memory bus. Such attacks recur because each generation's performance optimizations (deterministic encryption, speculation, shared caches, and interconnects) reopen an observable channel. Unlike a cryptographic assumption, which a scheme can strengthen on its own, a hardware-anchored design cannot close this channel without the vendor, and the vendor optimizes for performance first. We treat side channels as out of scope for this survey too, and current confidential-compute hardware materially raises the bar against software-only attackers. We still record the exclusion as the finding of § 9.3, because it falls hardest on exactly the TEE and hybrid families whose entire guarantee is the silicon.

The deployment-addressable gaps are more tractable. Attestation is only as strong as its configuration, so a deployment must pin the expected measurement and require a fresh, nonce-bound quote on each session; otherwise an attacker can replay a stale attestation, or, as TEE.Fail shows, forge one outright once the bus encryption is broken. Access patterns leak which corpus items a query touches, which Opal's ORAM answers in-band at a cost. Some confidential-GPU parts still leave HBM regions unencrypted, a coverage gap closed on current-generation hardware.

Insight 1. On the right hardware, near-plaintext confidential inference is a solved capability, gated only on owning a confidential accelerator. Its one residual risk is then uniform across the family and independent of the mechanism: a physical or microarchitectural attack on the silicon itself, a class the vendor threat model deliberately excludes (§ 9.3) and that no design here can close on its own.

5 Static Obfuscation

Static obfuscation transforms the data and/or weights so the server sees only values it cannot interpret [43, 54], with no trusted hardware or only a minimal amount. It is the cheapest family in our taxonomy: cost is near-plaintext and no trusted hardware is required (Tier 1). That low cost is also why its privacy claims deserve the most scrutiny. The obfuscation is *fixed*, and the adversary is the white-box server. It knows the obfuscation algorithm and its parameters, and it can passively collect the intermediary results of its own computation and use them to train an inverter model. On the deployment-readiness criteria, obfuscation leads on performance and is often near-lossless. Accuracy is not uniform across the family, however, and depends on how hard each scheme tunes its hiding (§ 5.2).

A second advantage is infrastructural. These schemes leave the model architecture intact, so they fit standard inference stacks far more readily than crypto or hybrid schemes, which require a custom serving path. The deployment trade-offs, including who must perform the one-time obfuscation and hold the secret, are examined in § 9.2.

5.1 Approaches

In every scheme the client applies a secret transform so the server computes on values it cannot interpret, then de-obfuscates the result. They differ in *what* they transform. Most either obfuscate the model weights (AloePri, ObfusLM, KV-Cloak) or transform only the input embeddings and run on a stock, unmodified model (SGT, OSNIP, Eguard); CodeCipher is a domain-specific outlier for source code. In the §2 taxonomy the trusted core is none or light (Tier 1, at most a TEE holding keys), and the adversary is an honest-but-curious, white-box operator. The observation surface is everything the scheme offloads to the untrusted accelerator: the obfuscated input and output, and, depending on the scheme, the offloaded attention and externalized KV-cache, plus the intermediate activations of the offloaded matrix multiplications.

ObfusLM [42] permutes and re-synthesizes the vocabulary, embedding, and head under a *provable* (k, ε)-anonymity guarantee (cluster size k, DP budget ε). This is a formal footing that is rare in this otherwise heuristic family. Because the vocabulary itself is permuted, it protects the *generated output* as well as the input, at the cost of fine-tuning the model on the obfuscated embeddings before serving. **AloePri** [43] is its covariant successor. It carries the secret token permutation beyond the embedding and head into the *remaining* weights, using invertible matrices and additive Gaussian noise, so the server runs stock inference with no fine-tuning. The client de-obfuscates the permuted response, so AloePri protects the output as well. Its hiding strength is governed by several secret parameters, not one: a keymat expansion h that widens the residual stream to $d_{\text{obs}}{=}d{+}2h$, the scale of the additive Gaussian noise, and a per-head permutation window. Each trades accuracy for hiding. The expansion in particular makes the obfuscation matrices more ill-conditioned, which amplifies floating-point error layer by layer and is what degrades accuracy at large h (§ 5.2). **KV-Cloak** [46] is narrower, securing only the KV-cache (the cached keys and values that encode the prompt) against an attacker who reads the externalized cache, e.g. via a GPU-memory leak. It transforms the cache as $K'{=}S\,\hat{P}\,(K{+}A)\,M$: persistent invertible matrices S, M fused offline into the attention weights, a one-time permutation $\hat{P}$ refreshed per PagedAttention block, and a beacon mask A that recovers $\hat{P}$ without storing it. It is lossless and adds $<$0.45% to prefill, and, covering the cache rather than the I/O channel, composes with an input/output scheme for end-to-end protection. The secret matrices S, M sit in a TEE for key custody, and attention runs on the GPU. KV-Cloak's per-block refresh of $\hat{P}$ is the structural escape of § 5.3, making it the family's outlier and a bridge toward the hybrids of §7; it splits no computation, though, so we treat it as an obfuscation scheme.

The input-transform schemes leave the weights untouched. **SGT** [54] (Stained Glass) adds learned, input-conditioned Gaussian noise to the embedding sequence, $\tilde{x}{=}x{+}\mathcal{N}(\mu_\theta(x), \Sigma_\theta(x))$. The noise model is trained to minimize the mutual information between the clean and obfuscated embeddings, which by a PAC-Privacy bound limits how much observing $\tilde{x}$ helps an adversary reconstruct x, while a separate utility loss keeps the frozen model's output unchanged. **OSNIP** [8] projects the input embedding onto an obfuscated semantic null space, near-orthogonal to the recoverable subspace, using a client-side encryptor network trained against the server LLM. **Eguard** [44] trains a projection network with a dual-level mutual-information objective aimed explicitly at Vec2Text inversion [49], driving inversion F_1 down to ≈4% while keeping >98% task accuracy. **CodeCipher** [40] is domain-specific (coding). It trains, by discrete-gradient search (snapping each update to a valid token), a token-to-token confusion map: a permutation of the embedding-matrix rows that replaces each code token with a different one, minimizing task loss while raising obfuscation (perplexity and edit distance) so the cloud LLM still returns the correct output. It is trained on a white-box proxy and transfers to black-box LLMs.

5.2 Performance and Accuracy

Read off the headline columns of Table 3, this family looks unambiguously cheap. Every scheme runs at ≈1× plaintext and reports near-lossless accuracy. Both numbers are *sweet-spot* readings of a hiding knob that trades security, accuracy, and performance against one another at the same time (Table 6). The nuance the headline hides is where each scheme pays.

Performance. Online cost is genuinely small. KV-Cloak adds less than 0.45% to prefill, AloePri is near-zero, and

the learned input transforms add only milliseconds at the client. Eguard is the one exception at inference, because it runs its extra projection network on every forward pass, for an inference overhead of about 1.7×. The larger costs are *offline*, and they split the family in two. The weight-transform schemes (AloePri, KV-Cloak) need only a one-time transform of the weights, computed in minutes with no training. The learned transforms instead pay a training bill, and in every case it is a small add-on network trained on top of the frozen pretrained model, not the model itself. SGT trains an input-conditioned transform over hours to days. OSNIP trains a client-side encryptor. Eguard trains a 24-layer projection encoder, which makes each training step 1.6 to 3.4× slower than the same step without the defense. ObfusLM is the heaviest, because it does not stop at an add-on: it fine-tunes the model itself on the obfuscated embeddings, which the paper measures at over ten hours of server time, and the fine-tuned weights are then tied to one checkpoint. The headline of near-zero *inference* overhead therefore hides a real setup cost for half the family. The infrastructural advantage is real: every scheme but KV-Cloak runs on *unmodified* inference software, pushing the extra work to a client-side tokenizer or embedder. AloePri serves as a re-weighted checkpoint on stock vLLM, and SGT, OSNIP, Eguard, and ObfusLM run the model unchanged while the client supplies the obfuscated embeddings or token indices. KV-Cloak alone must insert obfuscation operators into the attention path. None needs the custom serving runtime that crypto and TEE schemes require.

Accuracy. It is not free either, and the hiding knob is what sets it. KV-Cloak is exactly lossless, because its matrices cancel in the attention product, so its accuracy is independent of the security parameter. The other schemes face a privacy-utility curve. ObfusLM's (k, ε)-anonymity loses about 4% on classification and about 6% on generation, and degrades further as the anonymity set k grows. Eguard holds above 98% task accuracy, but only because its objective is tuned against inversion specifically. AloePri is the most sensitive case, and its accuracy is not set by one parameter: the keymat expansion h, the additive Gaussian noise, and the per-head permutation window all trade accuracy for hiding. The expansion is the sharpest of them: a larger h makes the obfuscation matrices more ill-conditioned, so floating-point error is amplified through the model's depth and accuracy falls off a cliff past a model-dependent value. AloePri's reported 0 to 3.5% loss is the reading at a tuned operating point, not a guarantee that holds as any of these knobs is pushed for stronger hiding.

This sensitivity is itself a deployment factor, and it separates the family by *how* the hiding configuration is found. The training-free schemes (AloePri, KV-Cloak) need no training, but a human must choose their hiding parameters for each model, and a safe configuration is easy to miss. Such a scheme is only as strong as how well that choice is formalized against the model's hyperparameters and the known attack vectors. A well-formalized heuristic approaches an information-theoretic guarantee, but few schemes formalize it, so the tuning is largely empirical today. The trained schemes (SGT, OSNIP, Eguard, ObfusLM) instead let an add-on network learn the hiding, which removes the manual-tuning risk but pays a heavy training cost and is itself retrained per model.

Insight 2. In static obfuscation, performance, accuracy, and security trade off against one another. A configuration strong on all three is attainable but non-trivial to find, and reaching it usually costs ground on at least one axis. Locating that configuration is itself a cost: the training-free schemes must choose their hiding parameters well, the trained schemes must train an add-on network.

5.3 Limitations and Known Attacks

Static obfuscation applies one secret transform and reuses it on every request. The attacks here are best read not as proof that obfuscation fails, but as a map of how hard each scheme has to work and of where that work runs out. They fall into two kinds (Table 7). The first kind a scheme can close or blunt with its own transform and knobs: a stronger, direction-changing transform, more noise, a null-space projection, or inversion-hardened training. The second kind is genuinely structural. Because the one secret is identical on every request, these attacks compound across requests, and no setting of the knobs removes them;

Table 6: The static-obfuscation trade-off is set by a hiding knob that moves all three axes at once. "≈1×, near-lossless" (Table 3) is the sweet-spot reading; this table shows what turning the knob up costs.

Scheme	Hiding knob	Accuracy cost as knob ↑	Performance cost
AloePri	keymat expansion h; Gaussian noise; per-head window	accuracy cliff (ill-conditioning compounds over depth)	wider residual stream d+2h (online)
SGT	MI-loss strength	−0.3 to 0.5 pp util	≈0 online; offline train (hrs to days)
OSNIP	null-space dimension	near-lossless	≈0 online; offline encryptor train
Eguard	MI objective / projection rank	>98% acc (inversion F_1→≈4%)	≈1.7× online; 1.6 to 3.4× train
KV-Cloak	block size b (key space $b!$)	lossless (decoupled)	<0.45% prefill
ObfusLM	cluster k, budget ε	−4% clf / −6% gen; worsens as k ↑	≈0 online; offline fine-tune (>10 h)
CodeCipher	PPL/obfuscation threshold	−3 pp Pass@1	≈0 online; offline opt.

Table 7: Attacks against static obfuscation. The mitigable attacks are closed or blunted by the scheme's own transform and knobs; the structural attacks compound across requests under the reused secret, and only refreshing it removes them (§7).

Attack	Target	Effect	Defense
Mitigable: the scheme's transform and knobs raise or close the bar			
Embedding / nearest-neighbour inversion [34, 41, 50]	learned embedding transforms (SGT, OSNIP)	recover input tokens from a single obfuscated embedding	inversion-hardened training (Eguard); null-space (OSNIP); large k (ObfusLM)
Deeper-layer inversion [16]	"deep layers are safe" assumption	inversion succeeds from deep and mid-level states	protect early layers too
Single-shot weight solving [42, 43, 55, 63, 71]	direction-preserving or permutation-revealing obfuscation (STIP, TransLinkGuard, SOTER)	direction similarity recovers >98% of weights; vocabulary-matching and precomputed-noise solve the static secret in one shot	direction-changing, noised transform (AloePri, ObfusLM), at an accuracy cost
Structural: compound across requests under the reused secret; only refreshing removes them			
Gram-matrix + ICA/BSS [5, 9, 28]	static linear / orthogonal mixing	$U^\top U{=}H^\top H$ leaks the covariance; fixed mixing makes BSS identifiable as observations accumulate	refreshed or non-orthogonal mixing
Learning-based inversion [16, 37, 49]	externalities offloaded under a fixed transform	a trained inverter learns the inverse from accumulated or synthetic data	refresh; reduce offloaded surface
KV-cache inversion / collision / injection [46]	externalized KV-cache; collision breaks the *fixed* shuffle (KV-Shield)	rebuilds the prompt from a leaked cache	KV-Cloak: fresh per-block permutation

the only fix is to stop reusing the secret, which moves a scheme out of the static family (§7).

Single-query inversion (mitigable). The learned embedding transforms are invertible from a single observation: the inversion attack on obfuscated embedding matrices [41] and nearest-neighbour attacks [34, 50] recover tokens directly, and inversion succeeds even at deep and mid-level layers [16]. Here the family's knobs genuinely help: inversion-hardened training (Eguard, $F_1{\to}{\approx}4\%$), a null-space projection (OSNIP), or a large anonymity set (ObfusLM's k) materially reduce recovery. AloePri's evaluation still reports prior obfuscation schemes losing more than 30% accuracy while leaking more than 50% of tokens, the trade-off the better-tuned schemes work to avoid.

Single-shot weight solving (mitigable by a strong transform). With the weights observable, a transform that only permutes or preserves vector directions is solvable from a single observation: direction similarity alone recovers more than 98% of such weights [71], and vocabulary-matching and precomputed-noise attacks reconstruct the secret outright on the lightweight schemes that leave directions intact (STIP, TransLinkGuard, SOTER, TSQP) [55, 63]. The schemes we report defeat this because they do more than permute: AloePri wraps its permutation in random invertible key matrices and additive noise, and ObfusLM in a (k, ϵ)-anonymity synthesis, which changes the very directions the attack relies on. The protection is heuristic, bought with accuracy rather than proven, but it holds at the recommended settings, where AloePri recovers fewer than 5% of tokens [42, 43].

Accumulation under the reused secret (structural). What no static transform closes is accumulation. Because the secret is identical on every request, observations compound. Under a static linear or orthogonal mixing the Gram matrix leaks the hidden-state covariance ($U^\top U{=}H^\top H$), and the fixed mixing makes blind source separation identifiable as observations pile up [5, 9, 28]. A trained inverter learns the inverse from the externalities the scheme offloads (hidden states, attention, KV-cache), all produced under the one transform [16, 37, 49]. The externalized KV-cache is reconstructible by inversion, collision, and injection, and the collision attack defeats a *fixed* shuffle (KV-Shield) precisely because the shuffle is reused [46]. The deterministic token map adds a standing residual, leaking token frequencies like a substitution cipher, which AloePri and ObfusLM suppress but cannot remove [42, 43].

The structural escape. The accumulation channel, where a trained inverter or blind source separation strengthens as observations pile up under the one secret, is closed only by *refreshing* that secret. Within this family only KV-Cloak does so, drawing a fresh permutation per PagedAttention block, which is why it alone withstands the collision attack; the hybrid-split schemes generalize the move with per-batch or per-offload refresh (§7).

Open Problem 1. *Re-obfuscation scheduling.* A fixed secret is cheap, but under it a trained inverter and blind source separation only strengthen as observations accumulate. Refreshing the secret resets that accumulation; refreshing on every request removes it but is costly. *Is there a refresh period longer than a single request, yet short enough that neither attack succeeds within one window, that holds recovery below a target at a fraction of the per-request cost?* A heuristic for the re-obfuscation frequency that minimizes accumulation under a latency budget is an open problem.

6 Differential Privacy

Differential privacy (DP) provides a formal but probabilistic guarantee by adding calibrated noise. The strength of the guarantee is set by the privacy budget ε: it bounds how much any single record can change the output distribution, so a smaller ε gives stronger privacy and requires more noise. Several schemes in other families borrow DP machinery: the covariant obfuscation of §5 adds Gaussian noise to weights and analyzes it under Rényi-metric DP, ObfusLM perturbs synthesized embedding weights under a Laplace mechanism, and the hybrid SCX of §7 injects Laplace noise into its KV-cache encoding. What sets the schemes in this section apart is not that they use noise but where the guarantee comes from. In the obfuscation and hybrid cases the noise is an adjunct to a secret key (a permutation, a one-time encoding) and is applied statically to the model's weights; protection ultimately rests on the secret, and if it leaks the noise alone does not protect the data.

The schemes here derive protection from the *randomness alone*, with no secret to keep: the bound holds against an adversary who knows the weights, the mechanism, and the budget, and the noise is drawn fresh per request and applied to the data or the output rather than embedded in the weights. The only cost is accuracy: a smaller ε injects more noise and lowers it.

Within the family, the approaches divide by *whose* data they protect and from *whom*, the confidentiality-scope axis of § 2.1.1: a scheme either protects the user's own input from an untrusted operator, or protects the operator's retrieval corpus and context from the consumer of the answer. These map onto the two standard settings of DP, local and central: the first noises the input on the client, the second noises the output on a trusted server. On the deployment-readiness criteria DP scores well on performance, near-plaintext at inference, and offers a formal threat model, trading accuracy for that guarantee (§9).

6.1 Approaches

The schemes agree on the primitive, calibrated noise under a budget ε, but differ on *which* DP notion they target and *where* the noise enters. The noise mechanism fixes the notion. Additive mechanisms cover the continuous case: the Gaussian mechanism yields (ε, δ)-DP with tight Rényi composition, while the Laplace mechanism and its generalized (metric) form yield pure ε-DP and metric-LDP, whose indistinguishability degrades with the Euclidean/Mahalanobis distance between inputs. The exponential mechanism instead handles discrete selection, where additive noise is meaningless, by sampling an output with probability that increases with utility.

Protecting the user's input (local DP). In local DP, the client perturbs its own data before sending it, so the operator is never trusted; the cost is that the client must run the encoder, moving part of the computation, and the trusted computing base, onto the device. Token-level local DP replaces discrete tokens through an exponential mechanism (SANTEXT, InferDPT); as a fixed client-side map it is one of the static input transforms of §5, and we treat it there. The three embedding-level schemes form a progression in *how much of the noise mechanism is trained* rather than fixed by a closed-form formula (each starts from a pretrained encoder and adapts it, never training from scratch); model retraining mirrors this inversely, heaviest where the noise is fixed and minimal where the noise is trained. **DP-Forward** [17] trains none of the noise: it adds a fixed analytic matrix-Gaussian calibrated only to the L_2 sensitivity (the largest Euclidean-norm change a single input can cause in the pre-noise layer output) of the pre-noise layers, a *task-agnostic* mechanism (the noise does not depend on the downstream task). It pays for that fixed noise with the heaviest model training of the three, fine-tuning the whole pipeline to perform on the noisy embeddings; the guarantee is standard (ε, δ)-DP, in its local sequence form (ε, δ)-SeqLDP (≈92% on SST-2 at ε≈16). **SPARSE** [65] trains *where* the noise goes: over a frozen encoder, a concept mask scores which embedding dimensions carry a user-specified sensitive attribute, and a Mahalanobis (elliptical Laplace) mechanism perturbs those dimensions more than the rest, under metric-LDP (keeping ≈65% utility at ε=10 while cutting leakage from 60% to 19%). **NVDP** [18] trains *the noise distribution itself*. It adds a single nonparametric variational information-bottleneck (NVIB) layer on a pretrained encoder, which fits a posterior over the multi-vector embedding and samples the shared embedding from it. The noise is thus calibrated to the task, and its Rényi-divergence bound converts to a Bayesian-DP guarantee (≈83% on MRPC at ε_μ≈11).

Each scheme thus occupies a different design point along these two axes, the DP notion and the trained part of the mechanism, with minimal model retraining except for DP-Forward, which absorbs its fixed noise wholesale. *The output is not hidden and gains no privacy of its own:* the operator returns it in the clear, and the noise costs it only accuracy. The perturbation protects the input, keeping the operator from inverting either the embedding or the output back to it.

Protecting the operator's context (central DP). The output schemes invert the trust assumptions. Here a *trusted* operator holds the private data and injects the noise server-side; the client adds nothing. They protect not the user's own input but the third-party data the operator serves at inference time, the RAG retrieval corpus or the documents assembled into the prompt (for example a company's confidential files or a patient's records), against the party that *consumes* the output. Because this data enters only at inference, training-time filtering does not address it, and their confidentiality scope (§ 2.1.1) differs from that of the local schemes above. **DP-KSA** [60] (the

DP-RAG generation method) treats the retrieval corpus as the private dataset: it draws an ensemble of candidate generations and releases a differentially private set of answer keywords through the propose-test-release framework with an exponential-mechanism count, so the answer is (ε, δ)-DP with respect to whether any one document is in the corpus. Its retrieval is ordinary dense top-k; the privacy is on the generated answer, not the query or the access pattern. **DP-Fusion** [62] bounds how much sensitive spans in the context can move the output: at each decode step it blends the next-token distribution computed with those spans toward one computed without them, until their Rényi divergence falls under the per-group budget, then samples from the mixture. The operator runs the model and sees the context in the clear, with no trusted hardware; the bound defends whoever *receives* the output, and applied per retrieved chunk it doubles as a prompt-injection defense. The two central schemes cover different operator-side assets: DP-KSA protects corpus membership, while DP-Fusion protects context spans.

Open Problem 2. *Composing local and central DP.* The two are orthogonal, protecting different assets on opposite sides of the trust boundary, so in principle a single pipeline could apply both, hiding the user's input from the operator and the operator's corpus and context from the consumer. Whether the two noise sources then compose without compounding the accuracy loss is untested: no surveyed work runs DP mitigations in sequence [7].

6.2 Performance and Accuracy

At inference every scheme here is near-plaintext in compute: the local schemes add one noise draw to the forward pass, the central schemes add sampling over the output. The real costs lie elsewhere and split along the local/central line. The local schemes pay *offline*, once: DP-Forward fine-tunes under the noise layer at roughly the cost of an ordinary fine-tune; SPARSE trains its concept mask in a one-time pass that scales linearly with data (25 min for 10k samples, under 45 min for 20k, on a single RTX 3090); NVDP trains one extra information-bottleneck layer on a pretrained encoder. Once trained, all three serve at $\approx 1\times$, and because post-processing is free under DP the noised embedding can be reused across tasks.

The central schemes pay *online*: DP-KSA multiplies generation by its ensemble size ($\approx N\times$ calls, with $N \approx 80$). DP-Fusion runs $m+1$ forward passes per token for m sensitive groups; because these passes parallelize, wall-clock can stay near a single pass, but only when compute and memory are available. Deployment needs an inference path that issues and fuses the parallel passes (the method is implemented as a wrapper around standard models, so it does not require a custom inference engine, but it does require that orchestration).

Within this family the three axes are only loosely coupled: privacy and accuracy always trade against each other through ε, whereas privacy and performance interact only on the central side. For the local schemes, turning ε down (more privacy) costs only accuracy, not compute, since the noise draw is the same size at any ε, and a more selective mechanism (SPARSE's masked noise, NVDP's learned posterior) costs only offline training. The central schemes trade the same way; here the answer degrades toward its *no-private-context* baseline (the answer the model would give without the corpus or context at all) as ε decreases: DP-KSA's accuracy drops to the no-retrieval baseline near $\varepsilon \approx 1$ and rises above it by $\varepsilon \approx 2$, while DP-Fusion's answer approaches the one it would give with the sensitive spans removed.

How steeply accuracy falls follows from each mechanism's ε-to-noise map: additive mechanisms (Gaussian, Laplace) scale noise as $\propto 1/\varepsilon$, a convex curve that is flat at moderate ε and collapses below $\varepsilon \approx 1$, with the collapse point set by the sensitivity of what is perturbed. Selection and blending mechanisms (the exponential mechanism, distribution mixing) instead saturate, rising with ε above a release threshold and flattening once the budget is loose.

6.3 Limitations and Known Attacks

The attacks on this family fall into two classes by whether DP can answer them (Table 8). Some are *mitigable* within the mechanism. The local embedding schemes are all targets of embedding inversion: Vec2Text [49] and GEIA [37] reconstruct text from representations, and DP-Forward in particular is recovered by a trained inversion model (IMA, over 75% token recovery in AloePri's evaluation) once ε is loosened enough to keep utility. Lowering ε or concentrating the noise on the sensitive dimensions weakens these attacks (SPARSE's concept-masked noise cuts Vec2Text leakage from 60% to 19% at $\varepsilon = 10$), so the inversion threat is managed on the ε-utility frontier rather than removed. DP-Fusion's gap is coverage, not mechanism: its bound holds only over spans a tagger marks sensitive, so a missed span falls outside the guarantee, and a higher-recall tagger closes it.

The remaining limits are *structural* and cannot be repaired with more DP. Two of the notions traded here are weaker than standard (ε, δ)-DP. Metric-LDP (SPARSE) only makes two inputs indistinguishable in proportion to their ℓ_2 distance in embedding space, so far-apart inputs can be separated with nearest-neighbour search. Bayesian-DP (NVDP) bounds the shift in an adversary's belief against an *assumed* data distribution, not the worst case. Its privacy is therefore only as sound as that assumption: a distributional mismatch, or a GAN-style reconstruction trained on the true distribution, weakens it. The ε-utility frontier itself is structural: at $\varepsilon \to 0$ the output is independent of the protected data, so there is a floor below which no scheme in this family can both protect and function.

Stacking mitigations across pipeline stages is likewise unresolved: the joint privacy guarantee and combined accuracy cost of running local and central DP together are

Table 8: Attacks on the differential-privacy family, by whether DP can answer them. Mitigable attacks are weakened by a smaller budget or by concentrating the noise; structural ones rest on a weaker notion or on the ε-utility floor and admit no purely-DP fix.

Attack	Target	Effect	Class / fix
Mitigable: a smaller budget or concentrated noise raises the attacker's cost			
Embedding / inversion-model [37, 43, 49]	DP-Forward, SPARSE, NVDP	reconstruct tokens from the noised embedding; IMA recovers >75% on DP-Forward at deployment ε	lower ε or concentrated noise
Tagger false-negative [62]	DP-Fusion	spans the tagger misses fall outside the bound	higher-recall tagger
Structural: a weaker notion or the ε-utility floor; no purely-DP fix			
Nearest-neighbour under metric-LDP [34]	SPARSE (metric-LDP)	indistinguishability scales with ℓ_2 distance, so far-apart inputs stay separable	inherent to the weaker notion
GAN reconstruction [27]	NVDP (Bayesian-DP)	the belief-shift bound assumes a data distribution; a generator trained on it erodes the budget	inherent to Bayesian-DP
Corpus membership inference [2]	DP-KSA	infer whether a document is in the corpus from the answers	DP is the defense; the ε-utility floor is the limit

both uncharacterized (Open Problem 2). One advantage holds despite all of these limits: central output-DP composes with any retrieval or inference backend, which makes it the family's natural answer to the corpus-extraction attacks of §9, regardless of how the rest of the pipeline is protected.

7 Hybrid Split: TEE + Obfuscation

The hybrid split keeps a small trusted core in a CPU-TEE (Intel TDX or AMD SEV-SNP) and offloads the heavy linear algebra to an *untrusted commodity GPU* over obfuscated activations, recovering the exact result in the enclave. On a performance-versus-threat spectrum it sits *below* both pure confidential-GPU inference (§4) and static obfuscation (§5) on raw speed. It pays a real cost for the TEE↔GPU communication it induces (§9), and some members trade a little accuracy to offload more. But it offers two advantages in return: stronger security than static obfuscation (the hiding randomness is refreshed, not fixed), and no dependence on the scarce, expensive confidential-GPU hardware that pure-TEE inference needs at scale. That trade, more security and commodity hardware at a performance cost relative to a confidential GPU, is the family's rationale; we substantiate where it lands among the families in §9, not here.

7.1 Approaches

Every scheme in this family makes the same two-part design choice: *what* to keep in the enclave, and *how* to hide the rest. A small trusted core (a CPU-TEE, or in GELO's case a confidential GPU) holds the secrets and runs the operations that are cheap but revealing: a secret transform and its inverse, the non-linearities, or merely the obfuscation keys. The bulk of the cost, the dense linear algebra of the QKV and output projections and the feed-forward general matrix multiplications (GEMMs), is offloaded to an *untrusted, commodity* accelerator that never sees plaintext; correctness is recovered exactly in the enclave, so unlike DP (§6) the family pays no accuracy cost. In the §2 taxonomy the TCB is the enclave alone, the adversary is an honest-but-curious, white-box operator, and the observation surface is the stream of obfuscated activations crossing the PCIe boundary.

The schemes differ in those two choices. **GELO** [5] draws a *fresh* secret invertible matrix A for each offloaded batch (a block of token hidden states H), forms the mixed $U{=}AH$, offloads the Q/K/V and output projections on U to the untrusted GPU, and unmixes by A^{-1} in the enclave. Because A is never reused, recovering H is a single-batch blind-source-separation (BSS) problem: the adversary must unmix the sources H from the single mixed observation $U{=}AH$ without knowing A, and a fresh matrix each batch denies it the repeated observations such separation normally needs. (GELO is the family's outlier on hardware: its trusted core is a *confidential* GPU, an H200-class TEE, and it offloads to a pool of cheaper untrusted commodity GPUs such as L40S, a deployment choice that maximizes throughput when confidential GPUs are scarce.) **TwinShield** [74] secret-shares *both* the linear and the non-linear layers across a CPU-TEE root and the accelerator: its OutAttnMult recasts the data×data attention multiply as additive computation with precomputed offline terms, and OutSoftMax offloads the exponentials. It adds U-Verify integrity checks (a Freivalds generalization, the first to verify the non-linear SoftMax) so the accelerator may even be *malicious*. **ObfuscaTune** [21] keeps only the low-parameter, non-linearity-bearing layers in the TEE (the input embedding, the output head, and the LayerNorm/dropout layers, ≈5% of the parameters, under 0.5% at the 405B scale) and offloads the obfuscated attention and MLP weights. Its obfuscation is a single *static* invertible matrix per weight without the per-batch

refresh, so its confidentiality holds only while the served model stays *proprietary*: without the plaintext weights the operator cannot solve for the transform. **SCX** [80] protects the KV-cache not by encryption but by *permutation*, using token- and feature-level permutations plus redundant embeddings and calibrated Laplace noise (yielding $(\varepsilon, 0)$-DP) with one-time keys per decoding step, so the cloud can neither recover the input nor complete the next token without the user's key. **CMIF** [78] is a TEE + DP hybrid: it shields only the embedding layer in a client-side CPU-TEE and, rather than encrypting activations (which would force every non-linearity back into the enclave and pay the round-trip per layer), it sanitizes the *input tokens* with an optimized Report-Noisy-Max mechanism (ε-DP token substitution), so the sanitized sequence runs entirely on the untrusted GPU with no per-layer return.

7.2 Performance

Table 9: Offloaded compute vs. TEE↔GPU round-trips (L layers, H heads, T generated tokens; structural estimate). "Per pass" is one forward pass (prefill or a decode step), "full gen" the whole T-token generation. Offload bases differ: TwinShield, ObfuscaTune, and GELO keep the non-linears (and GELO the FFN) in the enclave, so <100%, while SCX and CMIF run the whole model GPU-side and protect the *data* instead. SCX's key-sharing leaves only its first l_s and last l_e layers crossing per step ($O(L{+}T)$ overall).

Scheme	Offl. FLOP	per pass	full gen
TwinShield	≈87%	$(2H{+}5)L$	$(2H{+}5)L(1{+}T)$
ObfuscaTune	≈95%	$\sim 4L$	$\sim 4L(1{+}T)$
GELO	≈76%	$2L$	$2L(1{+}T)$
SCX	≈100%	$l_s{+}l_e$	$O(L{+}T)$
CMIF	≈98%	1	$1{+}T$

The reported numbers (Table 3) follow one trend: the more linear algebra a scheme can safely offload, the closer its latency approaches plaintext. But the published figures are *not* measured on a common footing, and two of them warrant caution. GELO's headline "≈76% offloaded at 20–30% overhead" is a *synthetic microbenchmark* of a *single* projection offload (random tensors shuttled between two processes, not an end-to-end model run), and that overhead is dominated (≈80%) by inter-process communication, not the masking. The overhead is also U-shaped in batch size: ≈20% only at an optimal batch, rising again for large batches as the $O(n^3)$ generation of the mixing matrix dominates; GELO also offloads only the attention Q/K/V/O projections (MLP offload is left to future work). The number should thus be read as a best-case lower bound, not a full-inference cost. TwinShield offloads ≈87% of total transformer-inference *FLOPs* (a compute ratio, not a latency one), reporting 5.4× over a TEE-only baseline in its verifiable configuration (including U-Verify), ≈6.7× with privacy only, and 4.0–6.1× / 4.9–7.7× over prior verifiable / private schemes. But offloading the non-linears is precisely what *multiplies* the round-trips: the masks do not commute with SoftMax, so TwinShield splits attention into *per-head* OutAttnMult and OutSoftMax calls and pays $(2H{+}5)L$ of them per forward pass (Table 9), the most of any scheme here. Its wall-clock latency is therefore governed by serialization, not by the 13% of FLOPs it retains. ObfuscaTune reports 1.5–4.3× over unprotected inference, and SCX is effectively free online (≈1×, 36 ms for Llama-7B) having moved its key setup offline. The output is exact for GELO/TwinShield/SCX and near-exact for ObfuscaTune (numerical error grows with the obfuscation matrix's condition number).

The binding constraint on this family is not the FLOP-offload ratio but the TEE↔GPU *serialization* it induces: each offloaded GEMM stalls until the enclave returns the preceding non-linearity, idling the GPU on the PCIe round-trip. Table 9 makes this structural: it is the offload *granularity*, not the FLOP fraction, that sets latency, and the schemes span two orders of magnitude. TwinShield's per-head offload is $O(HL)$ round-trips per token; GELO and ObfuscaTune are $O(L)$ (few, but a TEE-heavy

Table 10: Attacks against the hybrid-split family. Static-key schemes are broken; the surviving designs refresh their hiding randomness.

Attack	**Target**	**Effect**	**Defense / status**
Game of Arrows [71]	lightweight static weight-obfuscation (TransLinkGuard, SOTER)	>98% weight recovery via direction-similarity	per-batch refresh (GELO)
Hidden No More [63]	permutation + noise obfuscation, open-weights (STIP/PermLLM-class)	near-perfect prompt reconstruction from hidden states	refresh (GELO, SCX); seq.-sharding (Cascade)
Static-basis break [55]	reused secret basis (SOTER, TSQP)	confidentiality + integrity break; LLaMA-3-8B layer in ≈6 min	fresh keys; no static basis
Gram-matrix leak [5]	GELO under *orthogonal* mixing A	$U^\top U{=}H^\top H$ leaks hidden-state covariance (passive; narrows the BSS search)	non-orthogonal A, or shield vectors
ICA/BSS + anchor [5, 9, 28]	GELO per-batch mixing	hidden-state reconstruction; partial only with many in-batch anchors	fresh per-batch A; shields
KV-cache reconstruction [46]	externalized KV-cache (inversion / collision / injection)	rebuilds the user prompt from a leaked cache	KV-cache obfuscation (§5)

residual: GELO keeps the FFN and all non-linears in the enclave); SCX collapses decode to $O(1)$ per token by sharing intermediate-layer keys with the cloud ($O(L{+}T)$ overall); and CMIF reaches a single handoff by sanitizing the input so the whole model runs GPU-side. Adjacent confidential-compute work reaches the same point: the TEE↔GPU transfer dominates, and can only be hidden by overlapping it with computation rather than by cutting it [51, 59]. This pipelining and concurrency angle is largely unexplored for the hybrid-split family.

Insight 3. A high offload ratio does not yield low latency. The real bottleneck is the TEE↔GPU round-trip count: the schemes differ by two orders of magnitude in it, from TwinShield's $O(HL)$ per token (one per attention head per layer) to CMIF's single handoff, and even SCX, which offloads essentially all compute, spends 86–94% of its latency on TEE↔GPU memory movement rather than computation, leaving the GPU idle on the interconnect. Offload granularity, not offload volume, is the design variable.

Open Problem 3. *Scheduling around serialization.* For a single request the round-trip count is largely irreducible, set by the model's layer dependencies; the remaining lever is concurrency: overlap secure transfers with computation (PipeLLM's speculative pipelined encryption [59]), batch across requests where intra-request concurrency is capped, use speculative decoding, and re-partition the TEE/GPU boundary at runtime to keep round-trips off the critical path. *How far can a scheduler hide TEE↔GPU serialization behind concurrent work without weakening per-request refresh?*

The family's advantage is therefore economic: commodity untrusted GPUs, where pure-TEE inference needs a confidential accelerator (§4) and cryptography pays 100–1000× (§3); we rank it against the other families on the deployment-readiness axes in §9.

7.3 Limitations and Known Attacks

The family's security rests on *dynamic, never-reused* hiding randomness; once the secret is static and the weights are known, the open-weights solve returns (Table 10). Direction similarity recovers >98% of statically obfuscated weights [71], and a precomputed-basis attack drives a full confidentiality-and-integrity break, a LLaMA-3-8B layer in ≈6 minutes [55]; both were demonstrated on lightweight static schemes (TransLinkGuard, SOTER, TSQP). ObfuscaTune uses the same static, unrefreshed obfuscation and would inherit this exposure, but its threat model assumes a *proprietary* model that denies the operator the plaintext weights the solve needs; its confidentiality therefore stands on model secrecy rather than on refresh. The schemes that refresh (GELO's per-batch mixing, SCX's per-step keys) remove the question.

Even refreshed obfuscation leaks *second-order* structure. Under an orthogonal mixing A the Gram matrix is invariant ($U^\top U{=}H^\top H$), exposing the pairwise inner products of the hidden states. A non-orthogonal A, or appended high-energy "shield" vectors, blunt the resulting independent component analysis (ICA), BSS, and anchor attacks, but the guarantee is per-batch and collapses under key reuse [5].

The externalized KV-cache that these schemes leave outside the enclave for throughput is itself a reconstruction surface: inversion, collision, and injection attacks rebuild the prompt from a leaked cache [46], a threat made concrete by practical GPU-memory leaks such as LeftoverLocals [57]. The co-located TEE+GPU design also pays for encrypting the PCIe stream, with residual exposure from unencrypted high-bandwidth memory (HBM) on some confidential-GPU parts. Trust is honest-but-curious and white-box by default; only TwinShield's U-Verify extends to a malicious accelerator.

Open Problem 4. *Cheap second-order hiding.* The Gram-matrix leak above persists under per-batch refresh, and its known fixes each cost something: a non-orthogonal mixing forces an $O(n^3)$ inverse per batch and risks ill-conditioning, while high-energy shield vectors displace user tokens and lower throughput. *Is there a refreshable transform that closes second-order leakage at near-orthogonal cost without sacrificing throughput?*

8 Confidential Retrieval and Storage

The RAG retrieval and storage layer is a confidentiality problem distinct from inference: the threat is not *what* the server computes but *which* records a query touches and what the encrypted index reveals at rest, so its primitives are access-hiding and data-hiding rather than compute-hiding. They compose with the compute-hiding inference layer (§3) but are costed and attacked separately. We organize the layer by the retrieval *functions* that a confidential RAG stack performs in sequence, and report on the schemes that make them confidential.

A standard RAG exercises two of these functions. *Scoring similarity over a stored index* (§ 8.1) is made confidential either by distance-comparison-preserving encryption (DCPE), which encrypts the vectors so a standard index still ranks the ciphertext, or by homomorphic scoring, which keeps the similarity values themselves encrypted. *Hiding which record a query fetches* (§ 8.2) is secured by private information retrieval (PIR) or oblivious RAM (ORAM). A Graph/LightRAG stack [25] adds three more. *Constructing the knowledge graph* confidentially (§ 8.3) has no dedicated primitive: it reduces to confidential inference over the extraction step (§3–§7) plus encrypted storage. *Expanding adjacency and fanning out to source chunks* (§ 8.4) uses structured and searchable encryption with encrypted multi-maps (SSE/EMM), layered over ORAM where the access pattern must hide as well. *Running the whole traversal end-to-end* (§ 8.5) relies on a trusted execution environment (TEE) or heuristic anonymization. Several schemes serve more than one function; we place each

Table 11: Threat models for the retrieval and storage schemes of Table 12, on the axes of § 2.1.1, grouped by the confidential retrieval function each scheme serves (§8). The white-box/black-box attack-surface axis is inference-specific and omitted (a retrieval adversary sees an encrypted index and a query interface). *Behavior*: HbC=honest-but-curious; mal.=malicious server. *Trust anchor*: none=single untrusted server (computational crypto); k-srv=k non-colluding servers; TEE=trusted hardware. *Protects* and *Residual leak* are the two facets of confidentiality scope (asset hidden from the server vs. what it still observes): `ap`=access pattern; sp=search/query-equality pattern; cmp.=distance comparisons. *Basis*: crypto=cryptographic; PPE=property-preserving encryption (provable but leaky); DP=differential privacy; HW=hardware-enforced; heur.=heuristic. Side channels are out of scope for every row (§ 9.3).

Scheme	Behavior	Trust anchor	Protects	Residual leak	Basis
Similarity over a hidden index — § 8.1					
SAP/DCPE	HbC	none	query+corpus	all cmp.	PPE
CAPRISE	HbC	none	query+corpus	q–doc cmp.	PPE+DP
TRSE	HbC	none	query+corpus	`ap`	crypto
RAGtime-PIANO	HbC	none	query+corpus	`ct` only	crypto
RemoteRAG	HbC	none	query+corpus	DP query	crypto+DP
p^{2}RAG	HbC	2-srv	query+`ap`	result size	crypto
Access-pattern hiding — § 8.2					
Panther	HbC	none	query+`ap`	none*	crypto
PIR-RAG	HbC	none	query+`ap`	cluster	crypto
Pacmann	HbC	none	node `ap`	op-type	crypto
Compass	mal.	none	query+node `ap`	op-type	crypto
Adjacency expansion and graph storage — § 8.4					
XorMM	HbC; mal. (VXorMM)	none	adj. volume	sp	crypto
FLASH	HbC	none	adj. volume	sp+kw-pair	crypto
PeGraph	HbC	2-srv	corpus	`ap`+degree	crypto
GORAM	HbC	3PC	`ap`+structure	op-type	crypto
H_2O_2RAM	mal.	TEE	`ap`+degree	none*	crypto+HW
End-to-end confidential KG-RAG — § 8.5					
Opal	mal.	CPU+GPU-TEE	data+`ap`	op-type	crypto+HW
ARoG	HbC	none	entity identity	anon.	heur.
PrivGemo	HbC	none	graph+identity	subgraph	heur.

* effectively no leak beyond the operation occurring (fully oblivious fetch).

Table 12: Confidential retrieval and storage performance, grouped by the retrieval function each scheme makes confidential (§8); all mechanisms. Latency is end-to-end query time unless noted ("gt" = graph traversal); Comm./storage is per-query communication or index-storage overhead; utility is retrieval exactness. (S)SE = (structured) searchable encryption; PIR = private information retrieval; HE = homomorphic encryption; (VH-)EMM = (volume-hiding) encrypted multi-map; ORAM = oblivious RAM; DP = differential privacy. * relative only (no absolute latency reported); n/r = not reported. "billion-edge" denotes the largest graph tested, not a latency. "lossy; QA-pres." = retrieval is not exact (the subgraph is pruned by design) but downstream answer quality is preserved. Confidential KG construction (§ 8.3) carries no row: no surveyed scheme performs it.

Scheme	Stage	Primitive	Latency	Comm./storage	Utility
Similarity over a hidden index — § 8.1					
SAP/DCPE [22]	store,retrieve	DCPE (symmetric)	≈0 ms	≈1×	approx (β)
CAPRISE [76]	store,retrieve	conditional ADCPE	2339 vec/s (enc.)	≈1×; $k' \gg k$ (DP)	approx (β)+DP
TRSE [79]	retrieve,score	2-round SE + HE scoring	≈4.5 s @4K (scoring)	n/r	exact
RAGtime-PIANO [29]	retrieve,e2e	CKKS + lattice PIR	rel. only*	n/r	exact (ct)
RemoteRAG [12]	retrieve,score	DistanceDP + PHE top-k	0.67 s	n/r	100% recall (DP)
p^{2}RAG [48]	retrieve,score	2-server SS	rel. only*	$O(m+N)$ B/query	exact
Access-pattern hiding — § 8.2					
Panther [38]	retrieve	PIR+SS+GC+HE	18 s @10M	284 MB/query	exact
PIR-RAG [68]	retrieve	lattice PIR	16.8 s @5K	n/r	exact (cluster)
Pacmann [83]	retrieve,gt	batched PIR (graph-ANN)	−22% lat @100M	≈1.5–14 MB/query	90% ANN quality
Compass [86]	retrieve,gt	encrypted HNSW + ORAM	0.57–1.28 s	n/r	exact
Adjacency expansion and graph storage — § 8.4					
XorMM [70]	retrieve,gt	volume-hiding EMM (SSE)	≈5 ms/q; ≈170 ms @vol 2^{15}	≈1.2–2.5× stor.	exact
FLASH [73]	retrieve,gt	VH-EMM (+conj.)	≈1.8 ms/q; ≈90 ms @vol 2^{15}	near-opt. overhead	exact
PeGraph [72]	store,retrieve,gt	SSE + secret sharing	<1 s/query @multi-M	n/r	exact (ranked)
GORAM [20]	retrieve,gt	3PC sqrt-ORAM (reserve)	58 ms–36 s @1.4B	n/r; init <3 min	exact
H_2O_2RAM [82]	store,retrieve,gt	doubly-oblivious RAM (TEE)	≈1000× faster than O_2RAM	n/r	exact
End-to-end confidential KG-RAG — § 8.5					
Opal [30]	retrieve,gt,e2e	TEE + Ring-ORAM	2.32 s/q (1.57×)	ORAM (n/r)	KG-filt. +13 pp vs ANN
ARoG [52]	retrieve,gt	KGQA machine-ID anon	no crypto cost	n/r	SoTA KGQA (lossy)
PrivGemo [58]	retrieve,gt	dual-tower KG anon (HMAC)	no crypto cost	≈70% fewer nodes	lossy; QA-pres.

* p^{2}RAG reports 3–300× vs. PRAG; RAGtime-PIANO up to 40× latency and 323× communication vs. PIR-RAG and GraphRAG; neither gives an absolute query latency.

where its contribution is distinctive and cross-reference the rest.

By the paper's deployment-readiness lens, these schemes are performant in isolation but rarely self-sufficient: the no-trusted-hardware designs cover retrieval and storage, not generation, and shift the embedding, reranking, and traversal steps to the client (§9).

8.1 Confidential Similarity over a Hidden Index

Dense retrieval ranks the corpus by similarity to a query embedding. The confidentiality problem for this function is to compute that ranking without revealing the stored vectors or the query: embeddings are invertible to text [49], so the query embedding discloses the user's question and the stored vectors the corpus (Table 12). Two families make the similarity computable over hidden vectors and trade differently between leakage and cost.

Distance comparison preserving encryption (DCPE). The most direct approach encrypts each vector with a symmetric transform ($c = s\,e + \lambda$, the same dimension as the plaintext embedding, so the ciphertext imposes ≈1× storage) that preserves distance *comparisons*, so a standard similarity index ranks the ciphertext at near-plaintext speed with no interaction and no homomorphic operations (**CAPRISE** encrypts 2339 vectors/s). This is the drop-in confidential dense index. The two schemes differ in *which* comparisons survive. **SAP** [22] preserves *all* pairwise comparisons, including those among database vectors, which leaks the corpus's relational structure. CAPRISE [76] is *conditional*: it encrypts the stored vectors and the query under two different procedures, so only query-to-document comparisons survive and the distances among stored vectors do not. A differential-privacy perturbation of the query blocks inversion, closing the structural leak SAP leaves open. Either way the cost is exactness and a quantified comparison leak, examined in § 8.6.

Homomorphic and secret-shared scoring. Where any comparison leak is unacceptable, a second family keeps the similarity *values* themselves encrypted: the server scores the encrypted query against encrypted database vectors so neither the scores nor their order ever appear in the clear. **TRSE** pairs two-round searchable encryption with homomorphic top-k scoring; **RAGtime-PIANO** layers CKKS scoring over a lattice-PIR backend; and **p²RAG** secret-shares the database and the query to two semi-honest, non-colluding servers that jointly compute the scores. **RemoteRAG** is two-stage: it first retrieves a coarse set of k' candidates against a DistanceDP-perturbed embedding (the server sees only the noised query), then computes *exact* cosine-similarity top-k over those k' with the true query under partially-homomorphic encryption (PHE), reaching 0.67 s at 100% recall for a small privacy budget. These schemes prune the candidate set by DP or clustering before scoring, since the cost is the homomorphic arithmetic per candidate; the benefit is that a single mechanism hides both the query embedding and the at-rest vectors.

8.2 Access-Pattern Hiding

A scheme can hide the vectors and still leak *which* records a query touches or returns; over repeated queries this access pattern alone supports database-reconstruction attacks [32]. The schemes for this function close that channel: the server returns the matching records without learning which were touched. **Panther** and **PIR-RAG** retrieve through private information retrieval (single- or multi-server, classical or lattice-based), so the fetch reveals nothing about which record matched. The cost driver is the obliviousness itself: PIR keeps communication sublinear but forces a server-side linear scan, which is why these run in seconds at million-scale rather than milliseconds (Panther 18 s at 10M, PIR-RAG 16.8 s at 5K).

Compass and **Pacmann** reach the same goal over a graph-based approximate nearest neighbor (ANN) index (a Hierarchical Navigable Small World (HNSW) or Navigating Spreading-out Graph (NSG)): here the "graph" is a proximity index over document-chunk embeddings, each vertex a stored chunk vector rather than a knowledge-graph entity, so both are dense-retrieval schemes hiding the search over those vectors. Pacmann runs the iterative ANN traversal on the client and privately fetches each hop's neighborhood by batched PIR over a public database, retaining 90% of a non-private ANN run's accuracy while *cutting* latency by 22% at 100M vectors. Compass wraps an encrypted HNSW index in client-side Ring-ORAM to defend even a *malicious* server, leaving only the operation type visible, at 0.57–1.28 s even over a slow cross-region network. Opal (a TEE design, § 8.5) reaches the same access-hiding goal with hardware trust instead of client-side cryptography.

Insight 4. At Tier 1 (no trusted hardware), access-pattern hiding is not free; its cost moves to the client. Compass keeps the ORAM position map and runs a multi-round controller (≈500 MB of client state at corpus scale, against a 27–75× larger full client-side index), and Pacmann streams the entire database to preprocess it. The TEE alternative (Opal) hides the same access pattern with only a key and a counter on the client, but pays for it in a confidential GPU. The design variable is not whether obliviousness has a cost, but *where* it is paid: on the client or on trusted hardware.

8.3 Confidential Knowledge-Graph Construction

Graph/LightRAG retrieval presupposes a knowledge graph: an LLM extracts entities and relations from the corpus into a graph index before any query runs [25]. Constructing that index on untrusted infrastructure is the

one retrieval function no surveyed scheme makes confidential. It decomposes into two parts covered elsewhere: the extraction is itself an LLM inference over sensitive text, to which the inference families of §3–§7 apply, and the resulting graph must be stored encrypted (§ 8.4); but no design composes them into an outsourced confidential construction.

8.4 Adjacency Expansion and Graph Storage

Once the graph exists, LightRAG retrieval matches query keywords to entity and relation nodes (a dense ANN over entity embeddings, served by the schemes of § 8.1–§ 8.2), then expands one hop to neighboring nodes and edges and maps each entity to its source chunks. These graph-specific operations leak along structural channels that dense primitives do not address: the *node access pattern* identifies the query's subject entity, the *adjacency volume* is a degree signature that re-identifies nodes against a public graph even under encryption, and the `source_id` *fan-out* links a query to specific source documents.

These channels are closed by general-purpose encrypted-graph-database machinery applied to the RAG adjacency step. **XorMM** and **FLASH** encrypt the adjacency and keyword multi-maps and pad result-set size so the query-result *volume* does not leak (adjacency and `source_id` fan-out alike), at a storage overhead ranging from ≈1.2–2.5× (XorMM) down to near-optimal (FLASH, within a small constant of the volume-hiding lower bound [3]); FLASH's conjunctive variant, which answers multi-keyword (AND) queries, additionally reveals which keyword pairs are queried together, and XorMM's verifiable variant VXorMM detects a malicious server's forged or incomplete postings via aggregate MACs. **PeGraph** combines searchable encryption with secret sharing across two non-colluding servers for ranked adjacency queries over million-entity graphs in under a second. **GORAM** is the heavier-trust three-party computation (3PC) fallback for billion-edge federated graphs (58 ms to 36 s per query, with a one-time index setup under 3 minutes), and **H_2O_2RAM** provides the doubly-oblivious RAM layer, inside a TEE, on which such a graph store runs. What remains across all of them is the search/query-equality pattern, not the content, closed only by layering ORAM (§ 8.2) or rotating keys per session.

8.5 End-to-End Confidential KG-RAG

No surveyed system is a fully confidential graph-RAG pipeline; the purpose-built designs come closest but rest on trusted hardware or heuristic anonymization, not provable cryptography. **Opal** runs the whole pipeline obliviously inside a TEE via Ring-ORAM (2.32 s per query, 1.57× a non-oblivious run) with no accuracy loss from the obliviousness, but over a metadata-only identifier graph rather than a content-extracted one, so it is end-to-end confidential by hardware trust without being graph-RAG in the extraction sense. The graph-RAG systems instead anonymize a pre-built graph, shielding only what the remote model sees at no cryptographic cost: **ARoG** [52] maps entity names to machine identifiers so the remote LLM reasons over de-identified triples, reaching state-of-the-art knowledge-graph question answering (KGQA) at some utility loss; **PrivGemo** [58] adds structural de-uniqueness, a local "Hand" shipping the remote "Brain" only a pruned, HMAC-pseudonymized subgraph (≈70% fewer nodes, re-keyed per session). Both keep the raw graph in a trusted local component and leave confidential construction out of scope, providing *obfuscation* with unquantified residual leakage rather than a provable guarantee. The deployable options therefore remain the per-function schemes above.

8.6 Limitations and Known Attacks

The retrieval schemes trade utility or residual leakage for their speed, and the attacks exploit that leakage per function. Table 11 places every scheme on the security axes of § 2.1.1, and Table 13 lists the known attacks.

Similarity over a hidden index. The distance-comparison-preserving schemes (SAP/DCPE, CAPRISE) preserve distance *comparisons* between ciphertexts by construction, and that is the leak: property-preserving encryption is invertible by a distribution-aware adversary [50], and the recovered embeddings read as text via Vec2Text [49]. SAP also preserves inter-document comparisons, exposing the corpus's relational structure. CAPRISE defends by making the comparisons *conditional* (obfuscating inter-document distances) and perturbing the query under differential privacy, cutting Vec2Text reconstruction from BLEU 83 to 12 [76] at the cost of a broader retrieval ($k' \gg k$) that trades downstream precision. Distance-revealing search also leaks k-nearest-neighbor result geometry, which alone suffices to recover values [34] and is closed only by oblivious fetch with query DP (RemoteRAG). The homomorphic and secret-shared scoring schemes keep the values encrypted and avoid the class entirely, but are interactive: they return encrypted scores the client must decrypt and rank, complicating a fully outsourced pipeline.

Access-pattern hiding. The fetch-oriented schemes leak through *access pattern* and *volume*: generic reconstruction attacks exploit access-pattern and communication-volume leakage [32], and volume-only attacks defeat schemes that hide *which* record is touched but not *how many* [6]. The access-pattern attacks are mitigated using the oblivious schemes: PIR (Panther, PIR-RAG) genuinely closes the access-pattern channel, and Compass's ORAM hides which node is touched. Volume is a separate channel: closing it requires making the returned count independent of the query, not merely

hiding which record is touched. A fixed-size PIR fetch and a constant top-k leak no volume by construction; a variable result set (such as the clusters PIR-RAG returns) must be padded up to a fixed size, or its candidate budget fixed under the query DP of RemoteRAG and CAPRISE, so every query exposes the same count.

Adjacency expansion and graph storage. The adjacency and keyword multi-maps leak *volume*: the size of a postings list or one-hop result set. The same volume-only reconstruction applies [6], and volume-hiding EMMs (XorMM, FLASH) bound it by padding. What remains is the search/query-equality pattern, closed only by layering ORAM or rotating keys per session. A node's *access pattern* (which entity a query resolves to) and its adjacency *degree* (its neighbor count) form a structural fingerprint; matched against a public version of the graph, that fingerprint re-identifies the encrypted node, the same leakage-abuse reconstruction seen in the dense case [32]. ORAM (Compass), doubly-oblivious RAM (H_2O_2RAM), and 3PC sqrt-ORAM (GORAM) close it by making accesses oblivious: the server no longer sees which node is touched or how many neighbors it has, so there is no fingerprint to match.

End-to-end KG-RAG. The attack here is *structural re-identification*. Anonymizing the view strips entity names but leaves the graph's topology intact, and a node's neighborhood is often distinctive enough to act as a fingerprint that re-identifies the entity behind the identifier. Repeated queries sharpen this fingerprint, because the retriever explores the same node the same way each time, exposing the same neighbors and partial paths [58]. **ARoG** masks only names, leaving this structural channel open, and its stable identifiers further link a query's subject across sessions. **PrivGemo** closes this exact channel: it clusters the anonymized subgraph into supernodes to blur distinctive neighborhoods, and re-keys the anonymization per session against cross-session linkage. Both remain heuristic obfuscation rather than a provable guarantee: the coarsened view still exposes cluster-level connectivity, the residual leakage is unquantified, and the local component holding the raw graph is fully trusted.

9 Synthesis

This section draws the per-family results of §3–§8 together into three views: the three-way trade-off that locates each family (§ 9.1), per-use-case deployment recommendations (§ 9.2), and the cross-cutting findings and open problems that no single family section could state on its own (§ 9.3). The central finding is that no family dominates: which scheme is deployable depends on the application, on the constraint it can least afford to relax.

9.1 Design Space

Every family maximizes at most two of the three criteria and pays on the third, and the security basis axis of § 2.1.1 explains why the third differs so much. Cryptographic inference (§3) is exact and rests on a provable security basis, and pays one to three orders of magnitude in performance. A GPU-TEE (§4) matches that exactness at near-plaintext speed, but its basis is the silicon: the guarantee reduces to hardware whose side channels every threat model excludes, and to the availability of the accelerator that holds it. Static obfuscation (§5) is the cheapest, running at near-plaintext speed throughout, but rests on a heuristic basis and loses accuracy in proportion to how aggressively it hides. Differential privacy keeps that near-plaintext cost and adds a formal but probabilistic guarantee, paying again in accuracy through the budget ε (§6). The hybrid split (§7) recovers exactness and a basis stronger than static obfuscation on commodity hardware, and pays instead in the TEE↔GPU serialization that its offloading induces. The criterion a family gives up is forced by the basis it rests on, not incidental, so no single point on this surface dominates the rest. Which trade-off a deployment can accept is what selects the scheme: a real-time service is bound by its latency budget, a regulated workload by the security basis it must be able to prove, a cost-sensitive one by the hardware it already has, and a quality-critical one by the accuracy loss it can tolerate. The next section reads the surface through these constraints, use case by use case.

9.2 Deployment Readiness

The same trade-off applies differently in each use case, because the constraint a deployment can least afford to relax changes with the task. We give the recommendation use case by use case.

Generation is dominated by the cost of autoregressive decoding, so its binding constraint is latency. A confidential accelerator (§4) can be most attractive option for an operator that has access to one, running the full model at near-plaintext speed, but the hardware is scarce and few deployments own it. Where it is absent, static obfuscation leaves the architecture intact and runs on commodity GPUs with stock inference software at near-plaintext cost, at the price of a heuristic basis. The hybrid split is a weaker fit here than its commodity-hardware advantage suggests, because its TEE↔GPU round-trips, one per non-linear step and so multiplied across layers and generated tokens, dominate its decoding latency. MPC and FHE pay one to three orders of magnitude, which is hard to justify for single-user outsourcing; MPC's natural setting is instead *collaborative* inference among mutually distrusting data holders, such as joint analysis over disjoint institutional records, where no party can be trusted to aggregate and a cryptographic guarantee across parties is essential.

Embedding and reranking run models far smaller than a

Table 13: Known attacks on confidential retrieval, grouped by the function whose schemes they target (§8; confidential KG construction has no schemes and no row). Each row names the attack, the leakage it consumes, the schemes at risk, and the defending scheme/primitive. PPE = property-preserving encryption; EMM = encrypted multi-map.

Attack (cited)	Leakage exploited	Schemes at risk	Defense
Similarity over a hidden index			
PPE inversion [50] + Vec2Text [49]	ciphertext distance comparisons + aux. distribution	SAP/DCPE	CAPRISE (conditional ADCPE + query DP)
kNN query leakage [34]	k-NN result geometry	distance-revealing search	oblivious fetch + query DP (RemoteRAG); HE scoring (TRSE)
Access-pattern hiding			
Access-pattern reconstruction [32]	which records are fetched/returned	non-oblivious SE retrieval	PIR (Panther, PIR-RAG); HE scoring (TRSE, RemoteRAG)
Volume leakage [6]	result-set size	non-volume-hiding indexes	padding / DP candidate pruning
Adjacency expansion and graph storage			
Volume leakage [6] (bound [3])	adjacency / postings-list size	encrypted multi-maps	volume-hiding EMM (XorMM, FLASH)
Access-pattern / degree re-identification [32]	node access + adjacency degree signature	encrypted graph stores (PeGraph)	ORAM (Compass), doubly-oblivious (H_2O_2RAM), 3PC (GORAM)
End-to-end confidential KG-RAG			
Structural fingerprinting	unique subgraph topology of the exposed view	anonymized graph views	structural de-uniqueness (PrivGemo)

generative decoder, which changes the trade-off: the workload does not need a high-end accelerator, so paying for a confidential GPU is unnecessary. This is the hybrid split's natural setting. It keeps a small trusted core in a commodity CPU-TEE and offloads to untrusted commodity GPUs (§7), giving exact output without enterprise hardware, and the short, non-autoregressive workload makes its round-trip cost far less acute than in generation. Static obfuscation and local DP run at near-plaintext speed where a heuristic or probabilistic basis is acceptable, and a cryptographic encoder (SHAFT, Euston) is the choice only when an exact and provable guarantee is required.

Dense retrieval divides by what must be hidden. When the access pattern and the inter-document structure must stay hidden, the access-hiding schemes are the appropriate choice: private information retrieval (Panther, PIR-RAG) returns a record without revealing which one, and homomorphic scoring (RemoteRAG, TRSE) keeps the similarity values encrypted. When near-plaintext speed is the priority and a bounded, quantified comparability leak is tolerable, CAPRISE is the preferable option. Its conditional distance-comparison-preserving encryption hides the embeddings at roughly 1× cost, its query-side differential privacy blocks embedding inversion, and unlike SAP it conceals the inter-document structure (§ 8.1). The deciding question is whether that residual leak is acceptable for the corpus.

Graph retrieval splits by whether a cryptographic guarantee is required. Its only option at near-plaintext speed is heuristic anonymization (PrivGemo, ARoG), which pseudonymizes entities and prunes structure at no cryptographic cost, but is lossy and carries no provable guarantee. A scheme with a cryptographic guarantee instead pays overhead, and there the choice trades cost against which structural channel it closes (§ 8.4). Hiding which node a query touches requires client-side cryptography (batched PIR in Pacmann, an ORAM-wrapped index in Compass) or the hardware route of a TEE (Opal); the cryptographic route shifts cost and state onto the client, the TEE route onto trusted hardware. Hiding the adjacency volume requires a volume-hiding encrypted multi-map (XorMM, FLASH) at the cost of storage overhead. No single scheme closes all three channels simultaneously.

Open Problem 5. *No confidential pipeline from query to answer.* Confidential retrieval (RemoteRAG, § 8.1) and confidential generation (the inference families, §3–§7) exist separately, but no design covers all steps: a query-to-answer pipeline must compose private embedding, retrieval, rerank and generation schemes. *Which retrieval-generation compositions preserve their guarantees, and at what combined utility cost?*

Table 14: Deployment-frontier schemes across the inference-and-RAG pipeline, by mechanism family (§3–§8; one or two per cell). Inference stages (embed, rerank, generate) require server-side private computation; storage and retrieval mark any private-access scheme. "—" = no frontier scheme known to us.

Pipeline stage	Crypto/MPC	TEE	Obfuscation	DP	Hybrid split
Embed	Euston	Ascend-CC	—	—	TwinShield
Store (vectors)	CAPRISE/RemoteRAG	Opal	—	CAPRISE	—
Retrieve (ANN)	CAPRISE/RemoteRAG	Opal+ORAM	—	RemoteRAG	—
Rerank	TRSE	Opal	—	RemoteRAG	TwinShield
Generate	PermLLM/CryptoMoE	Nvidia-CC/PipeLLM	AloePri/KV-Cloak	DP-KSA/DP-Fusion	GELO/SCX
Graph traverse	Pacmann/XorMM	Opal/H_2O_2RAM	ARoG/PrivGemo	—	—

9.3 Cross-Cutting Findings

Client reliance, offline setup, and multi-tenancy. The trust-anchor axis of § 2.1 brackets the client by construction, since a client need not be named as its own trusted base, hiding how much work several schemes shift onto it. Cryptographic schemes make the user a compute party in 2PC or run PIR and ORAM retrieval from the client (§3, §8); covariant weight obfuscation (AloePri, KV-Cloak, ObfusLM) has the client re-key the model and hold the secret, so the checkpoint is per-client and one model cannot serve many tenants (§5); and local DP runs the model up to the noise-injection point on the client and draws the noise there, DP-Forward trading more client compute for more utility as that point moves deeper (§6, 17). The hybrid split mostly avoids this by hiding dynamically, except SCX, which keeps the client encoding each token and recovering the first and last blocks while the cloud computes the rest (§7). Retrieval adds its own load: the client embeds and perturbs the query, Compass holds an ORAM position map and runs a multi-round controller, and Pacmann preprocesses the database (§8). Two reasons account for this: structurally, local DP and covariant weight obfuscation can originate the secret only on the client, forgoing multi-tenancy; for performance, a client-side cache, compute party, or embedder lowers server cost or saves rounds. The TEE is the workaround, Opal hiding the same access pattern with only a key and a counter on the client and so trading the client's burden for trust in the hardware (§8).

Cryptographic and hybrid schemes force a custom serving path. Whether a scheme runs on an off-the-shelf inference server is a deployment cost the mechanism tables omit. A custom serving path is required whenever the secret is applied during inference, in one of two degrees. Cryptographic inference is the heavier: it replaces the model's operators, evaluating the non-linearities through secret sharing, oblivious transfer, or homomorphic arithmetic, so the server runs a secure-computation protocol in place of the model (§3). The hybrid split and KV-Cloak are lighter: the model's own compute still runs on the GPU, but a refreshable transform is interposed on the activations or the KV-cache and must be injected and unwound each request, so the serving path carries operators the stock stack lacks (§7, §5). A static transform needs neither: the secret is applied offline (static obfuscation's re-keyed weights, served on unmodified vLLM) or on the client (local DP's noised embeddings), so the server runs the stock model unchanged (§5, §6). A TEE is the middle case, running stock inference software inside a confidential VM with attestation (§4). Stock-server compatibility is one more advantage the static-transform families hold over the cryptographic and hybrid ones.

Confidential accelerators are scarce and enterprise-only. The pure confidential-accelerator path, a GPU- or NPU-TEE trusted end to end, is the most performant route that is also exact and near-plaintext in speed, but it trades the confidentiality problem for a hardware-procurement one (§4). The confidential accelerators our survey covers are the NVIDIA confidential-computing GPUs (the H100/H200/B200 class) and the Huawei Ascend NPU; the platform space beyond these is narrow and still forming, since confidential GPU support is recent, concentrated in high-end enterprise parts, and gained inline TEE-I/O only with Blackwell (§4). Trusting such an accelerator end to end also binds a deployment to one vendor's silicon and attestation. That cost is marginal for frontier generation, where a high-end accelerator is needed anyway and confidentiality is the only added requirement, but pure overhead for embedding or reranking, where a small encoder needs no high-end accelerator yet no cheaper confidential accelerator exists to match the smaller workload. The scarcity is what the hybrid split sidesteps by offloading to commodity untrusted GPUs (§7).

For heuristic schemes, a refreshed secret is what survives leakage accumulation attacks. The same failure recurs across three families. A single secret, reused on every request, lets the adversary accumulate the externalities the scheme offloads (the obfuscated activations, attention, and externalized KV-cache) until they recover what the secret was meant to hide. Fixed permutations are broken by prompt reconstruction in cryptographic inference (§3), static obfuscation accumulates into blind source separation and learned inversion (§5), and a reused KV-cache shuffle is broken by collision (§7). In every case the schemes that survive are the ones that refresh the secret: KV-Cloak draws a fresh permutation per cache block, GELO a fresh mixing matrix per batch, and SCX a one-time key per generated token. Refreshing is necessary but not sufficient. Under an orthogonal mixing the Gram matrix $U^\top U$ equals $H^\top H$ whichever matrix is drawn, so a fresh secret per batch still exposes the hidden states' pairwise inner products, which the blind-source-separation and anchor attacks of §7 exploit. How often to refresh under a latency budget is itself open (Open Problem 1).

Malicious-operator security is an added integrity layer, not a free upgrade. Most schemes here assume an honest-but-curious operator; defending one that may also deviate (drop, replay, or tamper) requires adding integrity, by one of two routes. The cryptographic route adds authenticated computation: information-theoretic MACs make secret-sharing MPC actively secure (the SPDZ line), TwinShield's U-Verify is a Freivalds-style check over a malicious GPU (§7), and Opal authenticates memory against a malicious host with a Merkle tree (§4). The hardware route adds attestation, as in the confidential accelerators (Nvidia-CC, Ascend-CC). Malicious resistance is thus not hardware-exclusive, yet the surveyed cryptographic schemes (SHAFT, Fission, PermLLM, CryptoMoE) take

neither: all are proved only semi-honest. MACs could secure their linear core, but the permutation-reveal optimizations that give them their speed lie outside what a MAC protects. A gap remains: malicious resistance is achievable cryptographically, yet no surveyed crypto scheme pays for it, and the families that reach it rely on integrity machinery or trusted hardware the semi-honest designs omit.

The field-wide side-channel blind spot. The residual-leakage axis of the taxonomy (Table 2) gives a uniform answer: *every* scheme places the microarchitectural, cache, timing, and physical side channels (power analysis and the memory bus among them) out of scope. GELO excludes accelerator-side leakage beyond raw memory reads [5], Opal enumerates the in-enclave channels it does *not* defend [30], and the rest omit them. This is a deliberate exclusion, and it falls hardest on the TEE and hybrid families, whose entire trust is the silicon. Their best-documented weaknesses are exactly microarchitectural and physical: transient-execution attacks of the Spectre and Foreshadow class [33, 67], and physical memory-bus interposition such as TEE.Fail, which extracts keys and forges attestation from the deterministic memory encryption of server TEEs. The vendors classify such physical attacks as out of scope, with no fix planned [61]. Individual attacks are patched on the platform cycle, but the class recurs because each generation's performance optimizations reopen an observable channel, and unlike a cryptographic assumption a hardware-anchored design cannot close it without the vendor.

Graph-RAG is the most privacy-underserved setting. No surveyed work delivers an end-to-end confidential graph-RAG deployment, and it is the setting the field, ours included, covers least. What this SoK offers is a per-component map of which surveyed primitive serves which part of the graph pipeline (§ 8.4): entity and relation ANN lookups by Compass, Pacmann, and CAPRISE; one-hop adjacency by the volume-hiding multi-maps XorMM and FLASH; graph storage by PeGraph and GORAM; access hiding by Opal (TEE) or a client-side ORAM index; and the cheapest, heuristic protection by the anonymization of PrivGemo and ARoG. One stage stays uncovered across the corpus: building the graph index itself under confidentiality, before any query runs. Confidential graph construction is the clearest open problem in this setting.

References

[1] Ahmad Al Badawi, Andreea Alexandru, Yuriy Polyakov, and Vinod Vaikuntanathan. SoK: Private LLM inference using approximate homomorphic encryption. Cryptology ePrint Archive, Paper 2026/935, 2026.

[2] Maya Anderson, Guy Amit, and Abigail Goldsteen. Is my data in your retrieval database? membership inference attacks against retrieval augmented generation. arXiv:2405.20446, 2024.

[3] Megumi Ando and Marilyn George. The cost of suppressing volume for encrypted multi-maps. *PoPETs*, 2022(4), 2022. doi: 10.56553/popets-2022-0098.

[4] Davide Andreoletti, Alessandro Rudi, Emanuele Carpanzano, Francesco Lelli, and Tiziano Leidi. Privacy-preserving LLM inference in practice: A comparative survey of techniques, trade-offs, and deployability. Cryptology ePrint Archive, Paper 2026/105, 2026.

[5] Anatoly Belikov and Ilya Fedotov. Good-enough LLM obfuscation (GELO). arXiv:2603.05035, 2026.

[6] Laura Blackstone, Seny Kamara, and Tarik Moataz. Revisiting leakage abuse attacks. In *NDSS*, 2020. doi: 10.14722/ndss.2020.23103.

[7] Andreea-Elena Bodea, Stephen Meisenbacher, Alexandra Klymenko, and Florian Matthes. SoK: Privacy risks and mitigations in retrieval-augmented generation systems. In *IEEE SaTML*, 2026. URL https://github.com/sebischair/SoK-RAG-Privacy.

[8] Zhiyuan Cao, Zeyu Ma, Chenhao Yang, Han Zheng, and Mingang Chen. OSNIP: Breaking the privacy-utility-efficiency trilemma in LLM inference via obfuscated semantic null space. arXiv:2601.22752, 2026.

[9] Jean-François Cardoso and Antoine Souloumiac. Blind beamforming for non-Gaussian signals. *IEE Proceedings F (Radar and Signal Processing)*, 140(6): 362–370, 1993. JADE: joint approximate diagonalization of eigenmatrices for blind source separation.

[10] Nishanth Chandran, Divya Gupta, Sai Lakshmi Bhavana Obbattu, and Akash Shah. SIMC: ML inference secure against malicious clients at semi-honest cost. In *USENIX Security*, 2022.

[11] Yuntian Chen, Tianpei Lu, Zhanyong Tang, Bingsheng Zhang, Zhiying Shi, Yuxiang Luan, and Zhuzhu Wang. SoK: Private transformer-based model inference. Cryptology ePrint Archive, Paper 2026/491, 2026.

[12] Yihang Cheng, Lan Zhang, Junyang Wang, Mu Yuan, and Yunhao Yao. RemoteRAG: A privacy-preserving LLM cloud RAG service. In *Findings of ACL*, pages 3820–3837, 2025.

[13] Marcin Chrapek, Marcin Copik, Etienne Mettaz, and Torsten Hoefler. Confidential LLM inference: Performance and cost across CPU and GPU TEEs. arXiv:2509.18886, 2025.

[14] Cloudera. The state of enterprise AI and modern data architecture. Survey of 600 IT leaders (1000+ employees), fielded by Researchscape, Apr–May 2024, 2024.

[15] Aritra Dhar, Clément Thorens, Lara Magdalena Lazier, and Lukas Cavigelli. ASCEND-CC: Confidential computing on heterogeneous NPU for emerging generative AI workloads. arXiv:2407.11888, 2024.

[16] Tian Dong, Yan Meng, Shaofeng Li, Guoxing Chen, Zhen Liu, and Haojin Zhu. Depth gives a false sense of privacy: LLM internal states inversion. In *USENIX Security Symposium*, 2025.

[17] Minxin Du, Xiang Yue, Sherman S. M. Chow, Tianhao Wang, Chenyu Huang, and Huan Sun. DP-Forward: Fine-tuning and inference on language models with differential privacy in forward pass. In *CCS*, 2023. doi: 10.1145/3576915.3616592. URL https://github.com/xiangyue9607/DP-Forward.

[18] Dina El Zein and James Henderson. Differential privacy for transformer embeddings of text with nonparametric variational information bottleneck. arXiv:2601.02307, 2026.

[19] European Data Protection Supervisor. TechSonar: Retrieval-augmented generation (RAG). https://www.edps.europa.eu/data-protection/technology-monitoring/techsonar/retrieval-augmented-generation-rag, 2024.

[20] Xiaoyu Fan, Kun Chen, Jiping Yu, Xiaowei Zhu, Yunyi Chen, Huanchen Zhang, and Wei Xu. GORAM: Graph-oriented ORAM for efficient ego-centric queries on federated graphs. *PVLDB*, 18(10):3601–3614, 2025. doi: 10.14778/3748191.3748218.

[21] Ahmed Frikha, Nassim Walha, Ricardo Mendes, Krishna Kanth Nakka, Xue Jiang, and Xuebing Zhou. ObfuscaTune: Obfuscated offsite fine-tuning and inference of proprietary LLMs on private datasets. arXiv:2407.02960, 2024.

[22] Georg Fuchsbauer, Riddhi Ghosal, Nathan Hauke, and Adam O'Neill. Approximate distance-comparison-preserving symmetric encryption. In *SCN*, 2022. doi: 10.1007/978-3-031-14791-3_6.

[23] Xinwen Gao, Shaojing Fu, Lin Liu, Zhuotao Liu, Yuchuan Luo, and Yongjun Wang. Euston: Efficient and user-friendly secure transformer inference with non-interactivity. Cryptology ePrint Archive, Paper 2026/046, 2026.

[24] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec)*, 2023.

[25] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. LightRAG: Simple and fast retrieval-augmented generation. In *Findings of EMNLP*, 2025. URL https://github.com/HKUDS/LightRAG.

[26] Husheng Han, Xinyao Zheng, Yuanbo Wen, Yifan Hao, Erhu Feng, Ling Liang, Jianan Mu, Xiaqing Li, Tianyun Ma, Pengwei Jin, Xinkai Song, Zidong Du, Qi Guo, and Xing Hu. TensorTEE: Unifying heterogeneous TEE granularity for efficient secure collaborative tensor computing. In *ASPLOS*, 2024. doi: 10.1145/3622781.3674168.

[27] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. Deep models under the GAN: Information leakage from collaborative deep learning. In *ACM CCS*, 2017. doi: 10.1145/3133956.3134012.

[28] Aapo Hyvärinen and Erkki Oja. Independent component analysis: Algorithms and applications. *Neural Networks*, 13(4–5):411–430, 2000. Independent component analysis; foundational blind-source-separation method.

[29] Antonia Januszewicz, Jiachen Zhao, Meng Jiang, and Taeho Jung. RAGtime-PIANO: Efficient secure remote RAG. Cryptology ePrint Archive, Paper 2026/231, 2026.

[30] Darya Kaviani, Alp Eren Ozdarendeli, Jinhao Zhu, Yu Ding, and Raluca Ada Popa. Opal: Private memory for personal AI. arXiv:2604.02522, 2026.

[31] Andes Y. L. Kei and Sherman S. M. Chow. SHAFT: Secure, handy, accurate, and fast transformer inference. In *NDSS*, 2025. URL https://github.com/andeskyl/SHAFT.

[32] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. Generic attacks on secure outsourced databases. In *ACM CCS*, 2016. doi: 10.1145/2976749.2978386.

[33] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. In *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[34] Evgenios M. Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. Data recovery on encrypted databases with k-nearest neighbor query leakage. In *IEEE S&P*, 2019. doi: 10.1109/sp.2019.00015.

[35] Ryan Lehmkuhl, Pratyush Mishra, Akshayaram Srinivasan, and Raluca Ada Popa. Muse: Secure inference resilient to malicious clients. In *USENIX Security*, 2021.

[36] Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. In *EUROCRYPT*, 2021. doi: 10.1007/978-3-030-77870-5_23.

[37] Haoran Li, Mingshi Xu, and Yangqiu Song. Sentence embedding leaks more information than you expect: Generative embedding inversion attack to recover the whole sentence. In *Findings of the Association for Computational Linguistics: ACL*, 2023.

[38] Jingyu Li, Zhicong Huang, Min Zhang, Cheng Hong, Jian Liu, Tao Wei, and Wenguang Chen. Panther: Private approximate nearest neighbor search in the single server setting. In *CCS*, pages 365–379, 2025. doi: 10.1145/3719027.3765190. URL https://github.com/AntCPLab/OpenPanther.

[39] Yang Li, Xinyu Zhou, Yitong Wang, Liangxin Qian, and Jun Zhao. Private transformer inference in MLaaS: A survey. *arXiv preprint arXiv:2505.10315*, 2025.

[40] Yalan Lin, Chengcheng Wan, Yixiong Fang, and Xiaodong Gu. CodeCipher: Learning to obfuscate source code against LLMs. arXiv:2410.05797, 2024. Unofficial implementation: https://github.com/bzz/codecipher.

[41] Yu Lin, Qizhi Zhang, Quanwei Cai, Jue Hong, Ye Wu, Huiqi Liu, and Bing Duan. An inversion attack against obfuscated embedding matrix in language model inference. In *EMNLP*, pages 2100–2104, 2024. doi: 10.18653/v1/2024.emnlp-main.126.

[42] Yu Lin, Ruining Yang, Yunlong Mao, Qizhi Zhang, Jue Hong, Quanwei Cai, Ye Wu, Huiqi Liu, Zhiyu Chen, Bing Duan, and Sheng Zhong. ObfusLM: Privacy-preserving language model service against embedding inversion attacks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

[43] Yu Lin, Qizhi Zhang, Wenqiang Ruan, Daode Zhang, Jue Hong, Ye Wu, Hanning Xia, Yunlong Mao, and Sheng Zhong. Towards privacy-preserving LLM inference via covariant obfuscation. arXiv:2603.01499, 2026. Unofficial implementation: https://github.com/sheng1feng/Aloepri.

[44] Tiantian Liu, Hongwei Yao, Feng Lin, Tong Wu, Zhan Qin, and Kui Ren. Eguard: Defending LLM embeddings against inversion attacks via text mutual information optimization. arXiv:2411.05034, 2024.

[45] Jinglong Luo, Yehong Zhang, Zhuo Zhang, Shiyu Liu, Ye Dong, and Haoran Li. Cryptography-based privacy-preserving large language models: A lifecycle survey of frameworks, methods, and future directions. *Artificial Intelligence Review*, 2025. doi: 10.1007/s10462-025-11466-6.

[46] Zhifan Luo, Shuo Shao, Su Zhang, Lijing Zhou, Yuke Hu, Chenxu Zhao, Zhihao Liu, and Zhan Qin. Shadow in the cache: Unveiling and mitigating privacy risks of KV-cache in LLM inference. In *Network and Distributed System Security Symposium (NDSS)*, 2026.

[47] Baihe Ma, Yanna Jiang, Xu Wang, Guangsheng Yu, Qin Wang, and Caijun Sun. SoK: Semantic privacy in large language models. arXiv:2506.23603, 2025.

[48] Yulong Ming, Mingyue Wang, Jijia Yang, Jie Xu, Zihan Wu, Cong Wang, and Xiaohua Jia. P^2RAG: Efficient privacy-preserving RAG service supporting arbitrary top-k retrieval. arXiv:2603.14778, 2026. URL https://github.com/myl7/p2rag.

[49] John X. Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M. Rush. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[50] Muhammad Naveed, Seny Kamara, and Charles V. Wright. Inference attacks on property-preserving encrypted databases. In *ACM CCS*, 2015. doi: 10.1145/2810103.2813651.

[51] Tushar Nayan, Ziqi Zhang, and Ruimin Sun. SecureInfer: Heterogeneous TEE-GPU architecture for privacy-critical tensors for large language model deployment, 2025. arXiv:2510.19979.

[52] Yunfeng Ning, Mayi Xu, Jintao Wen, Qiankun Pi, Yuanyuan Zhu, Ming Zhong, Jiawei Jiang, and Tieyun Qian. Privacy-protected retrieval-augmented generation for knowledge graph question answering. arXiv:2508.08785, 2025. URL https://github.com/nlpgm/arog.

[53] NVIDIA. NVIDIA secure AI with Blackwell and Hopper GPUs. Whitepaper WP-12554-001 v1.3, 2025.

[54] Jay Roberts, Kyle Mylonakis, Sidhartha Roy, and Kaan Kale. Learning obfuscations of LLM embedding sequences: Stained glass transform. arXiv:2506.09452, 2025.

[55] Abhishek Saini, Haolin Jiang, and Hang Liu. Vulnerabilities in partial TEE-shielded LLM inference with precomputed noise. arXiv:2602.11088, 2026.

[56] Benedict Schlüter, Supraja Sridhara, Andrin Bertschi, and Shweta Shinde. WeSee: Using malicious #VC interrupts to break AMD SEV-SNP. In *IEEE Symposium on Security and Privacy (S&P)*, 2024.

[57] Tyler Sorensen and Heidy Khlaaf. LeftoverLocals: Listening to LLM responses through leaked GPU local memory. Trail of Bits; CVE-2023-4969; arXiv:2401.16603, 2024.

[58] Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, Liming Zhu, and Wenjie Zhang. PrivGemo: Privacy-preserving dual-tower graph retrieval for empowering LLM reasoning with memory augmentation. arXiv:2601.08739, 2026.

[59] Yifan Tan, Cheng Tan, Zeyu Mi, and Haibo Chen. PipeLLM: Fast and confidential large language model services with speculative pipelined encryption, 2025. URL https://github.com/SJTU-IPADS/PipeLLM. ASPLOS 2025; arXiv:2411.03357.

[60] Tingting Tang, James Flemings, Yongqin Wang, and Murali Annavaram. Differentially private retrieval-augmented generation. arXiv:2602.14374, 2026.

[61] TEEFail. TEE.Fail: Breaking trusted execution environments via DDR5 memory bus interposition. https://tee.fail, 2025.

[62] Rushil Thareja, Preslav Nakov, Praneeth Vepakomma, and Nils Lukas. DP-Fusion: Token-level differentially private inference for large language models. arXiv:2507.04531, 2025. URL https://github.com/rushil-thareja/dp-fusion-lib.

[63] Rahul Thomas, Louai Zahran, Erica Choi, Akilesh Potti, Micah Goldblum, and Arka Pal. Hidden no more: Attacking and defending private third-party LLM inference. In *ICML (PMLR vol. 267)*, 2025. near-perfect prompt reconstruction via vocabulary matching (decoder injectivity); breaks STIP/PermLLM/Centaur fixed permutations. arXiv:2505.18332.

[64] Rahul Thomas, Louai Zahran, Sunoo Choi, Aravind Potti, Micah Goldblum, and Arka Pal. Cascade: Token-sharded private LLM inference. arXiv:2507.05228, 2025.

[65] Yu-Che Tsai, Hsiang Hsiao, Kuan-Yu Chen, and Shou-De Lin. Concept-aware privacy mechanisms for defending embedding inversion attacks. arXiv:2602.07090, 2026.

[66] Mehmet Ugurbil, Dimitris Mouris, Manuel B. Santos, José Cabrero-Holgueras, Miguel de Vega, and Shubho Sengupta. Fission: Distributed privacy-preserving large language model inference. In *IEEE CNS*, 2025. doi: 10.1109/CNS66487.2025.11194960. URL https://github.com/jimouris/curl.

[67] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution. In *USENIX Security Symposium*, 2018.

[68] Baiqiang Wang, Qian Lou, Mengxin Zheng, and Dongfang Zhao. PIR-RAG: A system for private information retrieval in retrieval-augmented generation. arXiv:2509.21325, 2025.

[69] Chenxu Wang, Junjie Huang, Yujun Liang, Xuanyao Peng, Yuqun Zhang, Fengwei Zhang, Jiannong Cao, Hang Lu, Rui Hou, Shoumeng Yan, Tao Wei, and Zhengyu He. SoK: Analysis of accelerator TEE designs. In *NDSS*, 2026.

[70] Jianfeng Wang, Shi-Feng Sun, Tianci Li, Saiyu Qi, and Xiaofeng Chen. Practical volume-hiding encrypted multi-maps with optimal overhead and beyond. In *CCS*, pages 2825–2839, 2022. doi: 10.1145/3548606.3559345. URL https://github.com/CDSecLab/XorMM.

[71] Pengli Wang, Bingyou Dong, Yifeng Cai, Zheng Zhang, Junlin Liu, Huanran Xue, Ye Wu, Yao Zhang, and Ziqi Zhang. Game of arrows: On the (in-)security of weight obfuscation for on-device TEE-shielded LLM partition algorithms. In *USENIX Security*, 2025. doi: 10.5555/3766078.3766093.

[72] Songlei Wang, Yifeng Zheng, Xiaohua Jia, and Xun Yi. PeGraph: A system for privacy-preserving and efficient search over encrypted social graphs. *IEEE Transactions on Information Forensics and Security*, 17:3179–3194, 2022. doi: 10.1109/TIFS.2022.3201392.

[73] Jiaojiao Wu, Jianfeng Wang, Shi-Feng Sun, and Xiaofeng Chen. Flash: Practical volume-hiding encrypted conjunctive multi-map with optimal overhead. *IEEE Transactions on Dependable and Secure Computing*, 22(6):7802–7817, 2025. doi: 10.1109/TDSC.2025.3600572.

[74] Jiaqi Xue, Yifei Zhao, Mengxin Zheng, Fan Yao, Yan Solihin, and Qian Lou. Securing transformer-based AI execution via unified TEEs and crypto-protected accelerators. arXiv:2507.03278, 2025.

[75] Jiaqi Xue, Xin Xin, Wei Zhang, Mengxin Zheng, Qianqian Song, and Minxuan Zhou. SoK: Can fully homomorphic encryption support general AI computation? a functional and cost analysis. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2026. doi: 10.56553/popets-2026-0066.

[76] Huanyi Ye, Jiale Guo, Ziyao Liu, and Kwok-Yan Lam. Efficient privacy-preserving retrieval-augmented generation with distance-preserving encryption. In *IEEE FLLM*, 2025. doi: 10.1109/fllm67465.2025.11391120.

[77] Itay Yona, Jamie Hayes, Ilia Shumailov, and Nicholas Carlini. Stealing user prompts from mixture of experts. arXiv:2410.22884, 2024.

[78] Honglan Yu, Yibin Wang, Feifei Dai, Dong Liu, Haihui Fan, and Xiaoyan Gu. Towards confidential and efficient LLM inference with dual privacy protection (CMIF). arXiv:2509.09091, 2025.

[79] Jiadi Yu, Peng Lu, Yanmin Zhu, Guangtao Xue, and Minglu Li. Toward secure multikeyword top-k retrieval over encrypted cloud data. *IEEE Transactions on Dependable and Secure Computing*, 10(4): 239–250, 2013. doi: 10.1109/TDSC.2013.9.

[80] Mu Yuan, Lan Zhang, Liekang Zeng, Siyang Jiang, Bufang Yang, Di Duan, and Guoliang Xing. SCX: Stateless KV-cache encoding for cloud-scale confidential transformer serving. In *SIGCOMM*, 2025. doi: 10.1145/3718958.3750509. URL https://github.com/yuanmu97/scx.

[81] Fei Zheng, Chaochao Chen, Zhongxuan Han, and Xiaolin Zheng. PermLLM: Private inference of large language models within 3 seconds under WAN. arXiv:2405.18744, 2024. URL https://github.com/zfscgy/PermLLM.

[82] Leqian Zheng, Zheng Zhang, Wentao Dong, Yao Zhang, Ye Wu, and Cong Wang. H_2O_2RAM: A high-performance hierarchical doubly oblivious RAM. arXiv:2409.07167, 2024.

[83] Mingxun Zhou, Elaine Shi, and Giulia Fanti. Pacmann: Efficient private approximate nearest neighbor search. In *ICLR*, 2025. URL https://github.com/wuwuz/Pacmann.

[84] Yifan Zhou, Tianshi Xu, Jue Hong, Ye Wu, and Meng Li. CryptoMoE: Privacy-preserving and scalable mixture of experts inference via balanced expert routing. In *NeurIPS*, 2025. URL https://github.com/PKU-SEC-Lab/CryptoMoE.

[85] Jianwei Zhu, Hang Yin, Peng Deng, Aline Almeida, and Shunfan Zhou. Confidential computing on NVIDIA Hopper GPUs: A performance benchmark study. arXiv:2409.03992, 2024.

[86] Jinhao Zhu, Liana Patel, Matei Zaharia, and Raluca Ada Popa. Compass: Encrypted semantic search with high accuracy. In *USENIX OSDI*, 2025. URL https://github.com/Clive2312/compass.