

A SYSTEMATIC REVIEW OF PRIVACY RISKS AND DEFENSES IN RETRIEVAL-AUGMENTED AND AGENTIC LARGE LANGUAGE MODEL SYSTEMS

Shaman Ali¹, Ahsan Ali^{*2}, Nimra Rajput^{*3}, Haroon Memon⁴,
Muzamil Shah⁵, Saad Ali⁶, Naeem Ahmed⁷, Sajjad Ahmed^{*8}, Muhammad Sufyan⁹,

^{*1,2,3,4,5,6,7,8,9}Information Technology Centre, Sindh Agriculture University, Tandojam, Sindh, Pakistan

DOI: <https://doi.org/10.5281/zenodo.22972624>

Keywords

Large language models ·
Retrieval-augmented generation ·
LLM agents · Privacy leakage ·
Membership inference ·
Differential privacy · Prompt
injection · integrity

Article History

Received: 01 June 2026

Accepted: 23 August 2026

Published: 26 September 2026

Copyright: Shaman Ali,
Muzamil Shah, Haroon
Memon

Corresponding Author: *
Ahsan Ali, Nimra Rajput,
Sajjad Ahmed

Abstract

Large language models are no longer used as stand-alone text generators. In real-world situations, they become part of systems that pull documents from private data sources, track past chats, and use external tools to act on a user's behalf. This shift changes where personal information is stored and how it might leak. Previous privacy studies looked mainly at what a model remembers during training; less focus has been on leaks from search index embedding storage, agent memory, and tool connections. This review fills that gap. We group research published from 2023 to 2026 into a three-level system that splits risks into model data (L1), the search component (L2), and acting agents (L3). For each level, we look at how an attack works, what access the attacker needs, and how strong the proof is, using examples like copying search content at a 90 % success rate for adding information, five fake passages recovering 92 % of short texts from their embeddings, and models sharing too much context in 25–57 % of test cases. We then show which protections, like privacy in training and search, deleting data, fake data sets, limiting access, keeping control and data separate, and using hardware, match the threats they handle. We measure the cost to performance when numbers are available. The study shows that no single method covers all risks and that the weakest-protected risks appear when the whole system works together. We ended with a look at how to evaluate these systems, how to link them to rules, and a research plan for keeping privacy safe in complex AI systems.

1 INTRODUCTION

I remember when the first privacy question about an LLM was whether it had memorized something it should not have. That privacy question still matters today. Carlini and colleagues (2021) showed that GPT-2 would copy parts of its training data, including names, phone numbers, and email addresses. Later work by Carlini and colleagues (2023) showed that memorization grows with LLM size, data duplication, and prompt length. Nasr

and colleagues (2023) went further. Nasr and colleagues (2023) recovered training data from a production chatbot by forcing the chatbot into a "divergence" state, where the chatbot emitted memorized text about 150 times faster than normal. I see how the systems people use today differs from the models studied in those papers. A customer service assistant first retrieves passages from a company's documents before answering. A clinical assistant reads a patient's record through a

retrieval pipeline. A personal agent keeps a long-term memory of its user's preferences, reads the user's inbox, and sends emails through an API. Each of these parts retriever, vector store, knowledge base, memory module, tool interface – stores or moves information, and each creates a path by which that information can leave the system. I call these deployments compound LLM systems. I understand how the privacy consequences of this change are only beginning to be understood. Retrieval was first seen as a privacy improvement because facts could be kept outside the LLM and deleted on request. Zeng and colleagues (2024a) confirmed part of that

intuition: adding retrieval dramatically reduced the leakage of training data from an LLM.

That same study, however, showed that the retrieval database itself leaked easily, with hundreds of dialogues and email fragments reproduced verbatim from a hundred crafted prompts.

I notice similar patterns have since been reported for embedding stores (Morris et al. 2023) Agent memory (Wang et al. 2025), and tool-using assistants that can be hijacked by instructions hidden in the data they read (Greshake et al. 2023; Zhan et al. 2024

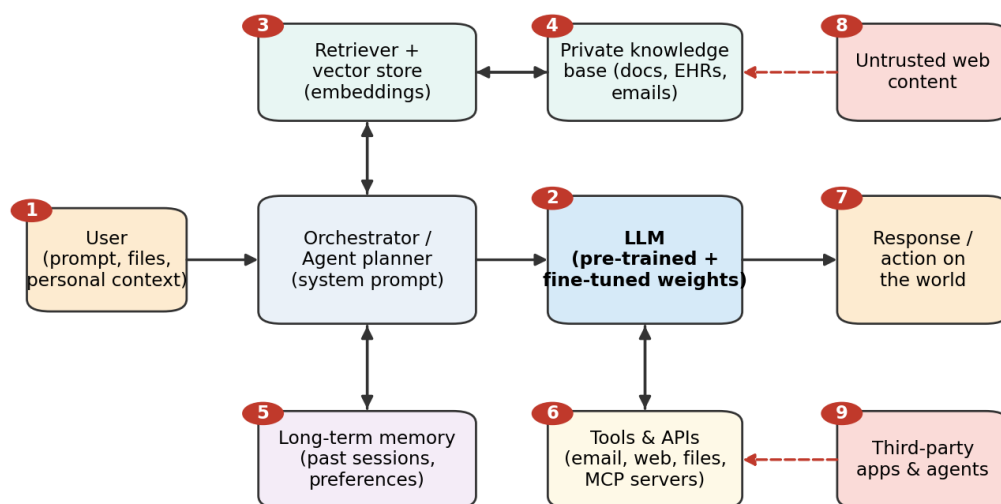


Fig. 1 Reference architecture of a compound LLM system and its nine principal privacy attack surfaces. Solid arrows show normal data flow; dashed red arrows show untrusted content entering the system. Numbers correspond to the attack surfaces discussed in Sections 5–7.

Existing surveys have catalogued LLM privacy threats thoroughly. Most of them keep the model at the center of the analysis. Das et al. (2025) Separate security from privacy attacks; Yao et al.

(2024) divide findings into uses, offensive uses and vulnerabilities; Yan et al. (2024) and Chen et al. (2025A) organize risks into leakage and attacks. Review defenses such as federated learning and confidential computing. These foundations are valuable. They do not, however, explain how privacy fails when the model is just one part among many, or compare defenses based on the specific system-level threats they neutralize.

This survey sets out to supply that account. Its contributions are as follows.

- It introduces a three-layer taxonomy model (L1), retrieval (L2) and agent (L3)—that places each privacy threat at the component where sensitive data is located, not based on the attack method (Section 4).
- It combines evidence from more than fifty studies published mainly between 2023 and early 2026, pulling out reported success rates to compare how serious each risk is while clearly noting limitations in comparing across studies (Sections 5–7).
- It matches eight families of defense to eight threat classes. It identifies which threats have the least protection coverage (Section 8) and summarizes the utility cost of protection when published data allow.
- It questions how evaluation is done, ties controls to the GDPR and the EU AI Act requirements and sets out a research plan for privacy-preserving compound systems (Sections 9–11).

The rest of the paper is structured like this. Section 2 explains the review method. Shows where this survey fits compared to previous work. Section 3 introduces the system model, the privacy ideas that are used throughout, and the model of the adversary. Section 4 provides a taxonomy. Sections 5, 6, and 7 examine model-layer threats, retrieval-layer threats, and agent-layer threats, respectively. Section 8 covers defenses. Section 9 discusses evaluation; Section 10 covers regulation and governance; and Section 11 highlights open problems. Section 12 ends the paper.

2 Review method and related surveys

2.1 Scope and search strategy

The review looks at privacy in language model-based systems that involve at least one of these features: retrieval augmentation, persistent memory, or tool use. We included model-level studies only when they help explain what happens at the system level or provide a defense that operates at the system level. We focused on work published between January 2023 and March 2026. Earlier foundational papers were also included to give context.

We reviewed papers from security conferences such as IEEE S&P, USENIX Security, and ACM CCS. We also reviewed papers from machine learning events, like NeurIPS ICLR, ICML, and COLM. We looked at natural language processing conferences like ACL and EMNLP and their Findings sections too. We also used arXiv. We used search terms like retrieval-augmented generation, RAG, agent, memory, tool, privacy, leakage, extraction, membership inference, embedding inversion, injection, unlearning, and differential privacy.

We filtered candidate papers based on two requirements: the work had to address the confidentiality of proprietary data, and it had to present empirical results or a clear mechanism. We then used reference lists and benchmark papers to do forward snowballing. Each study that passed these checks was coded by layer, adversary access, target asset, and, when relevant, defense family (see Figure 2).

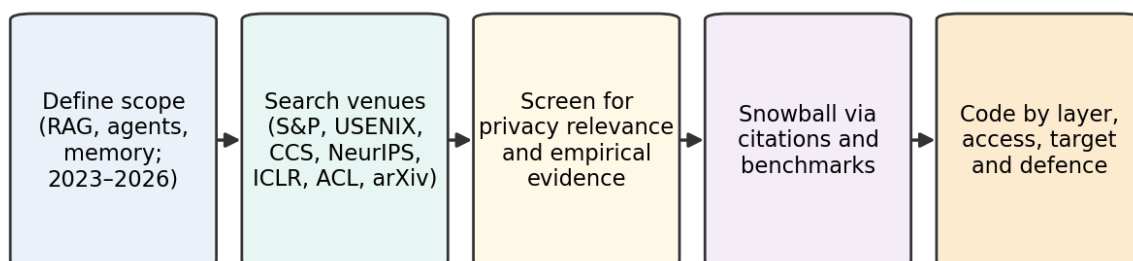


Fig. 2 Review workflow used to identify and code the literature.

Two limitations of this process deserve mention at the outset. First, much of the work exists only as preprints because the field moves faster than peer review. We indicate the publication status of each study in the reference list. Treat unreviewed results with corresponding caution. Second, reported success rates come from models, datasets and threat models, so they indicate orders of

magnitude rather than supporting direct ranking. The process will be revisited in Section 9.

2.2 Relation to earlier surveys

Table 1 compares the coverage of this survey with the coverage of six reviews.

The comparison is qualitative and reflects reading of every paper's main body, not just passing mentions.

Table 1: Coverage of this survey compared with earlier reviews (● covered in depth, ◐ partial, ○ mentioned or absent)

Survey	Model memorization	RAG / KB leakage	Embedding inversion	Agent memory	Injection-based exfiltration	Contextual integrity	Defense-threat mapping
Neel and Chang (2023)	●	○	○	○	○	◐	○
Yao et al. (2024)	●	○	○	○	◐	○	◐
Yan et al. (2024)	●	◐	○	○	○	○	◐
Das et al. (2025)	●	◐	○	○	◐	○	◐
Chen et al. (2025a)	●	○	○	○	○	◐	◐
He et al. (2025)	◐	◐	○	◐	●	◐	◐
This survey	●	●	●	●	●	●	●

Closest in spirit is the agent-security survey by et al. (2025). That survey discusses privacy alongside many other security threats that affect agents. This new survey is different in two ways. First, privacy stays at the heart of everything from start to finish. Second, it dives into the retrieval layer, where most enterprise deployments keep their sensitive data, and it treats that layer with the same level of detail as the agent layer. I notice that privacy is the focus throughout, and retrieval depth matches agent depth.

3 Background and threat model

3.1 Compound LLM systems

Retrieval-augmented generation combines a parametric language model with a memory. At inference time, a retriever selects passages from a corpus, and Retrieval-augmented generation conditions answers on passages (Lewis et al. 2020).

In one implementation, documents are split into chunks. Each chunk is mapped to a vector by an embedding model, and the vectors are stored in a similarity index. A user query is embedded in space. The nearest chunks are. Retrieval-augmented generation inserts the chunks into the prompt, along with system instructions.

Agentic systems extend this loop. The ReAct pattern interleaves reasoning steps with actions such as searching the web or querying an API Yao et al 2023 Modern agents add persistent memory records of past interactions that are retrieved much like RAG passages and a growing catalogue of tools. The Model Context Protocol, introduced in late 2024, standardized how agents connect to external tool servers and data sources Anthropic 2024 which has accelerated the integration of LLMs with e-mail, file systems, calendars and enterprise databases

From a privacy standpoint, the key observation is that the prompt the model sees at any moment is assembled from several sources with different owners and trust levels: the system developers' instructions, the user's request, retrieved documents, stored memories, and tool outputs that may contain text written by strangers. The model has no reliable way to tell these sources apart once they are concatenated, which is the root of several of the attacks discussed below.

3.2 Privacy notions

Three notions of privacy appear often in literature. Are used throughout this survey.

Extraction. A model memorizes a training example when it starts producing that example more often than when it produces a similar example that was not part of the training. Extraction attacks aim to recover those examples exactly as they appear. Membership inference attacks on the hand do not try to get the exact example. Instead, they only want to know whether a certain example was part of the training set or retrieval set (Shokri et al. 2017).

Differential privacy. A randomized algorithm is (ϵ , δ)-private if changing one record in its input changes the chance of any result by no more than a factor of e^ϵ plus a tiny extra amount δ (Dwork

and Roth 2014). Differential privacy (DP) limits what anyone can learn about a record, no matter what attack they use. In language models, it is often used during training with DP-SGD, which limits the size of each example's gradients and adds carefully chosen noise (Abadi et al. 2016).

Contextual integrity. Nissenbaum (2004) said privacy is broken not when information is made public, but when it moves in a way that violates the rules of the situation where it was shared. A message told to a doctor can properly go to a specialist but not to an employer. Contextual integrity is very important for agents, which often move information from one situation to another, from an inbox to a reply, from a record to a request, for a time slot, and must decide which moves are right.

3.3 Adversary model

Table 2 summarizes the adversary capabilities assumed by the attacks reviewed in this survey. We distinguish four levels of access. Most of the retrieval-layer and agent-layer attacks need only black-box query access, which is exactly what makes them feasible: a user of a public chatbot or anyone who can put text on a web page that the agent will read already has that black-box query access. These adversary capabilities show the threat is real, not just theoretical.

Table 2: Adversary capabilities assumed by the attacks reviewed

Access level	What the adversary can do	Typical actor	Example attacks
Black-box query	Send prompts and observe text outputs only	Any user of a public chatbot or RAG application	KB extraction, RAG MIA, MEXTRA, prompt extraction
Grey-box	Also observes token probabilities, retrieved sources or embedding outputs	API customer; plug-in developer	Min-K% Prob, LM inversion, Vec2Text
Content injection	Places text in data the system will later read (web page, e-mail, shared document, corpus)	Any third party; malicious insider	Indirect injection, PoisonedRAG, AgentPoison, Morris-II worm

Access level	What the adversary can do	Typical actor	Example attacks
White-box / infrastructure	Reads model weights, gradients, or memory of the serving host	Model thief; compromised cloud operator	Gradient inversion, weight inspection (addressed by TEEs)

We consider three classes of protected assets. Training data is the class of protected assets, and it includes pre-training and fine-tuning corpora. Deployment data is the class of protected assets and includes the retrieval corpus, the embedding index, system prompts, and agent memory. User data is a class of protected assets and includes prompt content, uploaded files, and information the agent can access through tools. A system can fail on any of these classes of protected assets while still being sound on the others, which is why a layered analysis of protected assets is useful.

4 A layered taxonomy of privacy threats

Fig. 3 shows the taxonomy that organizes Sections 5–7. The main idea is where the sensitive data sits when it gets exposed. This is not accidental. If we organize it by attack technique, as earlier surveys did, we end up grouping threats that share one fix. For example, taking data from weights and taking data from a knowledge base both mean we must limit how much a single query can reveal. In this way, threats that need different answers get grouped together. By organizing by location, we match the taxonomy to the real engineering choices a system designer needs to make: which part to protect and how much it will cost.

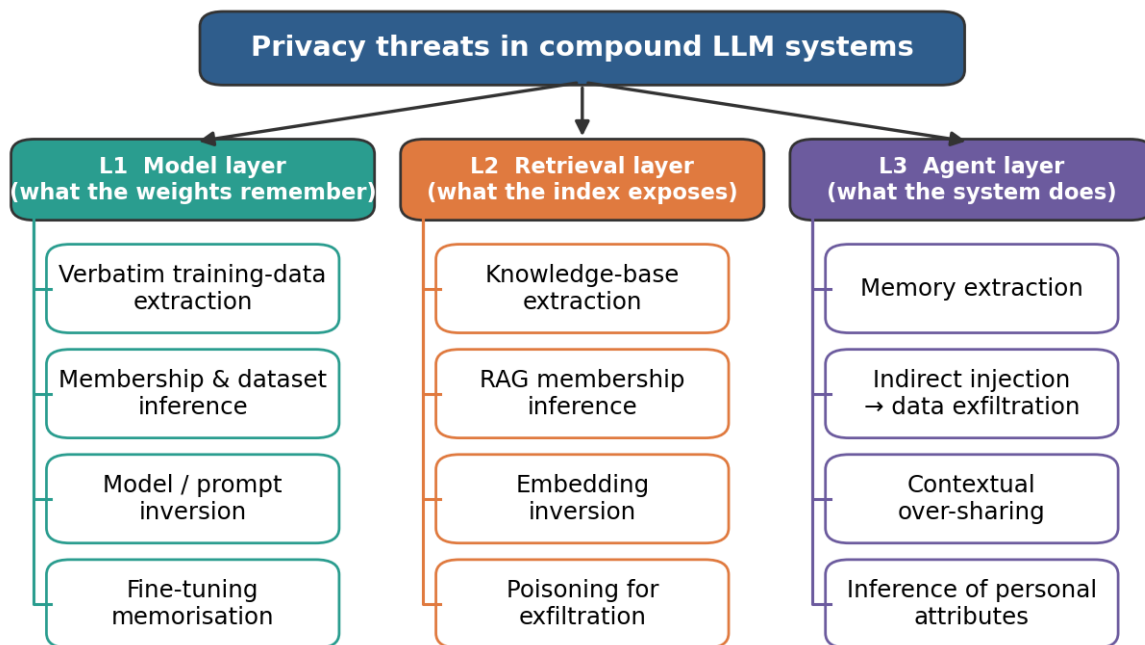


Fig. 3 Three-layer taxonomy of privacy threats in compound LLM systems. L1 threats concern information encoded in model weights; L2 threats concern the retrieval corpus and its embeddings; L3 threats concern information handled by the agent at run time.

The three layers are not independent. A poisoned document in the retrieval layer (L2) can carry instructions that cause an agent (L3) to send data to an attacker. A model tuned on retrieved content (L1) may memorize that content. When these cross-layer chains happen, we discuss the attack based on where it starts and note what happens downstream.

Figure 4 shows the studies on a timeline. Two patterns stand out. Research on attacks against retrieval and agents picked up quickly from 2024 onward, closely following real-world commercial use. Defenses came later, with a six- to twelve-month delay. Many of the defenses published in 2025 are benchmarks or design principles rather than actual tools that can be deployed.

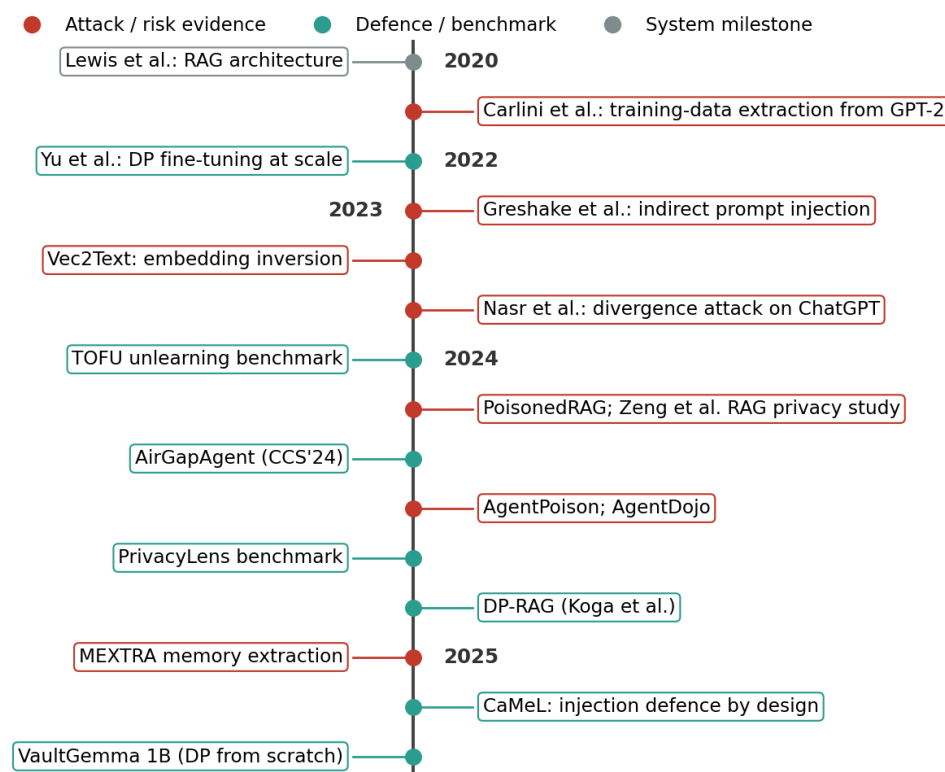


Fig. 4 Timeline of selected studies on privacy in LLM systems, 2020–2025. Attack and risk evidence (red) has consistently preceded defenses and benchmarks (teal).

5 Model-layer threats (L1)

5.1 Verbatim extraction of training data

The main threat in the model layer is about taking information: getting a model to repeat a training sequence exactly as it was. Carlini and others in 2021 created samples from GPT-2, sorted them using signals like how likely they were to appear, and found hundreds of remembered sequences, some including personal contact information. Their next study looked at what causes remembering. They found it grows in a near-linear

way as the model gets bigger, as an example appears more often in the training data, and as the prompt used to retrieve it gets longer. The real-world effect is troubling: the same things that make models better also make them more likely to remember.

I read that alignment was sometimes thought to close this channel because a chat model trained to be helpful is unlikely to recite raw web text. Nasr and colleagues in 2023 proved otherwise. By asking a production chat model to repeat one

word, they made the model abandon its chat mode and emit training data about one hundred fifty times faster than normal. They estimated that many gigabytes of data could be recovered from semi-open models with existing methods. The lesson, for system designers, is that alignment changes the model's default behavior but does not remove content from the model's weights.

5.2 Membership inference

Membership inference asks a question that data extraction does: was this record used in training? But it can still cause harm. For example, just being part of a dataset might reveal someone's medical condition. Shi et al. (2024A) introduced Min-K% Prob, a method that scores a text based on the log-likelihood of its least likely tokens. They also created the WikiMIA benchmark to test how well models can detect training data.

Fu et al. (2024) focused on tuned models. They improved membership inference by using a reference set generated by prompting the target model itself. This calibration strengthens and stabilizes the signal.

A careful large-scale study has lowered expectations. Duan et al. (2024) tested membership inference attacks on models ranging from 160 million to 12 billion parameters. These models were trained on the Pile dataset. In some cases, the attacks did only slightly better than random guessing. The researchers said this is due to two factors: very large training datasets and training that lasts for roughly just one epoch. Also, there's a line between what counts as member text and non-member text. They found that some reported successes were errors caused by shifts in data distribution between samples collected at different times.

Two clear conclusions come from this work. First, for -trained base models, membership inference attacks are currently not a serious threat. Second, for tuned models that train on small datasets over

multiple epochs, the situation is different. In that case, the attack surface stays wide open. Fu et al. (2024) showed this clearly. Their findings suggest caution when working with fine-tuned models.

5.3 Inversion of prompts and hidden instructions

System prompts often hide logic and sometimes hold credentials or internal policy text. Zhang et al. (2024) showed that simple text-based attacks can recover System prompts with high probability across eleven models, including commercial systems, and that a classifier can distinguish genuine extractions from hallucinated ones with high precision. Morris et al. (2024) demonstrated a route: reconstructing a hidden System prompt from the model's next-token probability distribution alone.

Because many applications place retrieved passages and user data inside the System prompt as developer instructions, prompt inversion can also expose deployment data as intellectual property.

5.4 Memorization introduced by fine-tuning

Organizations are increasingly customizing models using their own data—things like support tickets, clinical notes, legal documents and this is where the real risks, at the model level, tend to show up. Lukas et al. (2023) examined how identifiable information, or PII, can leak from fine-tuned versions of GPT-2. They found that removing PII before training helped reduce the risk. Didn't stop it completely. They also tried private training, which offered better protection, but that came at the cost of lower model performance. Meanwhile, Kim et al. (2023) took an angle focusing on the data subject. They created ProPILE, a tool that lets people check if a model has memorized their personal details. Users can build prompts using information they already know about themselves. See if the model reveals anything sensitive.

Table 3 summarizes the model-layer studies reviewed here.

Table 3: Representative model-layer (L1) privacy studies

Study	Venue	Threat	Access	Main finding
Carlini et al. (2021)	USENIX Sec.	Training-data extraction	Black-box	Hundreds of memorized GPT-2 sequences recovered, some with PII
Carlini et al. (2023)	ICLR	Quantifying memorization	White-box	Memorization grows log-linearly with size, duplication and context
Nasr et al. (2023)	arXiv	Extraction from aligned model	Black-box	Divergence attack raises emission of training data about 150×
Shi et al. (2024a)	ICLR	Pre-training data detection	Grey-box	Min-K% Prob; WikiMIA benchmark
Duan et al. (2024)	COLM	MIA evaluation	Grey-box	MIAs near random for large pre-trained LMs; shift artefacts
Fu et al. (2024)	NeurIPS	MIA on fine-tuned LMs	Grey-box	Self-prompt calibration strengthens membership signal
Zhang et al. (2024)	arXiv	System-prompt extraction	Black-box	Prompts recovered with high probability across 11 models
Morris et al. (2024)	ICLR	Prompt inversion from logits	Grey-box	Hidden prompts reconstructed from next-token distributions
Lukas et al. (2023)	IEEE S&P	PII leakage after fine-tuning	Black-box	Scrubbing reduces but does not remove PII leakage
Kim et al. (2023)	NeurIPS	PII probing by data subjects	Black-box	ProPILE lets individuals test for their own PII

6 Retrieval-layer threats (L2)

Retrieval moves data out of the model's weights and into a separate storage system. That storage is checked every time a request comes in. This setup makes it easier to delete data and control access. It also means that when an attacker controls the query, sensitive text can end up directly in the model's context window. Sometimes the text appears exactly as it was stored. Four types of threats have been shown to exploit this behavior.

6.1 Extraction of the knowledge base

Zeng et al. (2024A) did the first detailed measurement. They used a medical conversation dataset called Healthcare Magic, which has 200,000 talks between doctors and patients. They also used the Enron e-mail dataset as a source. They made 250 attack prompts. Each prompt had two parts. One part was information meant to guide the search. The other part was a command telling the model to repeat what it saw. Figure 5(a) shows their results. With Llama-7b-Chat general prompts, 117 parts of the search results were

copied exactly. Specific prompts found 89 conversations from Healthcare Magic and 107 personal information items from Enron.

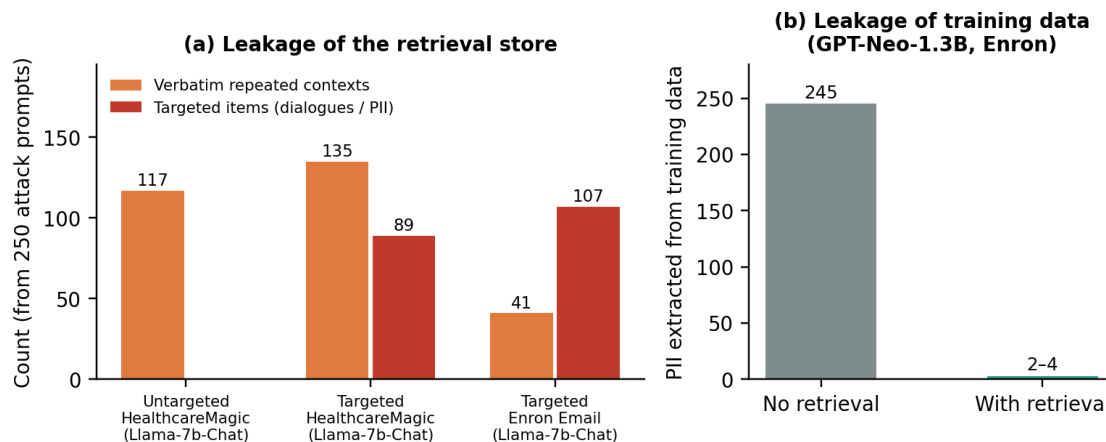


Fig. 5 Privacy effects of retrieval reported by Zeng et al. (2024a). (a) Leakage from the retrieval store under 250 attack prompts. (b) Leakage of training data from GPT-Neo-1.3B with and without retrieval augmentation, showing that retrieval shifts rather than removes risk.

Fig. 5(b) shows the other side of the same study. When retrieval was added to a GPT-Neo-1.3B model, targeted extraction of PII from the model's training data fell from 245 items to between two and four. Retrieval, in other words, does not so much reduce privacy risk as relocating it: the training data becomes harder to reach, and the retrieval store becomes easier.

I see that subsequent work has made extraction more scalable. Qi et al. (2024) showed that instruction-following ability itself is a vulnerability: an injection that asks the model to reproduce the model's context succeeded against all 25 randomly chosen custom GPTs. Each of those custom GPTs needed two queries, and the prompt injection recovered 41% of a 77,000-word document with 100 queries. The effect grew with model size.

Jiang et al. (2024) automated the process with an agent that reasons about feedback from each response and generates queries to explore unseen parts of the store. The agent extracted more than 70% of knowledge based on commercial platforms

and improved on earlier black-box methods by 45%

6.2 Membership inference against retrieval stores

Even when full extraction is prevented, an attacker might still want to know if a specific document is in the store. For example, they could be checking whether a patient's record has been indexed by a hospital assistant. Anderson et al. (2025) demonstrated that this can be done in both black-box and grey-box settings. They did this by prompting the system to say whether a given passage appears in its context. Li et al. (2024) took a different path. They noticed that documents stored in the system often generate responses with high semantic similarity to the query. Using this behavior, they developed the S²MIA attack. This method outperformed five baseline membership inference attacks. Managed to bypass three well-known defenses. Liu et al. (2025) used another approach. They masked words in the target document and then observed how accurately the RAG system was able to fill in the missing parts. This is only possible if the original document was retrieved and used during the generation process. But all these attacks have a flaw: their queries stand out. They look suspicious. Can be caught by standard query filters. Naseh et al. (2025) solved

this by creating a version. Their method asks sounding questions. The answers to these questions are only found in the target document. Because the queries look normal, they are hard to detect with filtering systems. This pattern follows a path in web-application security. Every time a detection rule is introduced, an attacker quickly comes up with an attack designed to look harmless. It's a cat-and-mouse game.

6.3 Embedding inversion

A common assumption in deployment guides is that storing embeddings rather than raw text offers some protection. Morris et al. (2023) showed that this assumption is mostly false. Their Vec2Text method treats inversion as controlled generation: it proposes a text, embeds it, compares the result with the target vector, and iteratively corrects the hypothesis. It recovered 92% of 32-token inputs exactly and reconstructed full names from embeddings of clinical notes. The consequence is

that a vector database containing embeddings of sensitive documents should be treated with the same protection as the documents themselves, and that sharing embeddings with third-party services is equivalent to sharing the text.

6.4 Poisoning as a route to exfiltration

A common idea in deployment guides is that keeping embeddings of the original text gives some sort of protection. Morris et al. (2023) showed that this idea is largely untrue. Their Vec2Text method looks at inversion to generate text on purpose. It suggests that a piece of text is turned into an embedding, checked for closeness to the target vector, and then adjusted to move closer. It was able to find 92% of 32-token inputs and even got full names from embeddings of medical notes. The result is that a database that holds embeddings of documents needs to be protected just like the actual documents. Sharing embeddings with companies is the same as sharing the original text.

Table 4 summarizes the retrieval-layer studies.

Table 4: Representative retrieval-layer (L2) privacy studies

Study	Venue	Threat	Access	Main finding
Zeng et al. (2024a)	Findings ACL	KB extraction	Black-box	Verbatim medical dialogues and PII extracted; retrieval lowers training-data leakage
Qi et al. (2024)	arXiv	KB extraction via injection	Black-box	25/25 custom GPTs leaked within 2 queries; 41% of a 77k-word text in 100 queries
Jiang et al. (2024)	arXiv	Adaptive extraction KB	Black-box	>70% of KBs on commercial platforms; +45% over prior methods
Anderson et al. (2025)	ICISSP	RAG membership inference	Black/grey-box	Prompt-based MIA; template instructions partly effective as defense
Li et al. (2024)	arXiv	RAG membership inference	Black-box	S ² MIA beats five MIAs and evades three defenses
Liu et al. (2025)	WWW	RAG membership inference	Black-box	Mask-and-fill signal reveals membership
Naseh et al. (2025)	CCS	Stealthy RAG MIA	Black-box	Natural questions evade query-level detection

Study	Venue	Threat	Access	Main finding
Morris et al. (2023)	EMNLP	Embedding inversion	Grey-box	92% of 32-token inputs recovered exactly; names from clinical notes
Zou et al. (2025)	USENIX Sec.	Knowledge poisoning	Injection	90% success with 5 texts per question in million-scale KB
Chen et al. (2024)	NeurIPS	Memory / KB backdoor	Injection	>80% average success at <0.1% poison rate; <1% benign impact

7 Agent-layer threats (L3)

Agents turn text generation into action. They read and write files, send messages, browse the web and remember users across sessions. The privacy risks in this layer arise less from what is stored than from what the agent *does* with it.

7.1 Extraction of agent memory

Agents that remember interactions usually keep them as records that they retrieve like RAG passages, and they are vulnerable in similar ways. Wang et al. (2025) proposed MEXTRA, a black-box attack that creates prompts to force an agent to retrieve and display records from its memory, and they also provided a way to build such prompts at different levels of attacker knowledge. They showed that typical agents can leak users' memory, and they examined how memory size, retrieval depth, and the similarity function affect leakage. Memory is a tempting target because it collects personal details, preferences, health information, and relationships users share with a system they believe is their own assistant.

7.2 Indirect prompt injection and data exfiltration

Greshake et al. (2023) introduced prompt injection, a technique where hidden instructions are placed inside content that a large language model will later read. This content can be a web page, an e-mail or a shared document. Because a model cannot reliably separate these hidden instructions from data, it may follow the planted instructions while acting with the same rights as

the user. Greshake et al. showed that this can lead to data theft, self-propagation, and application manipulation, including in a chatbot integrated with a search engine.

Cohen et al. (2024) turned self-propagation into a worm. An adversarial prompt hidden inside an e-mail causes a retrieval-augmented generation (RAG) assistant to repeat that prompt in its messages. This spreads the content to other assistants while also exfiltrating stolen data.

Two benchmarks now measure this threat in a way. I find that Injec Agent has 1,054 test cases built on 17 user tools and 62 attacker tools. The attacks are split into harm and private data theft. When a React-prompted GPT-4 agent tested 24% of the cases, the agent compromised the agent in the base setting. The compromise rate nearly doubled when the injected instruction was reinforced with a hacking prompt (Zhan et al. 2024). Agent Dojo provides 97 tasks, including email handling, banking, and travel booking, along with 629 security test cases. Agent Dojo is designed to grow as attacks and defenses appear (Debenedetti et al. 2024).

7.3 Contextual over-sharing

Not all agent leakages need an attacker. Agents often share information in unsuitable ways because they don't consider the rules that apply in each situation. Miresghallah et al. (2024) created the ConfAIde benchmark based on contextual integrity theory. They found that GPT-4 and ChatGPT gave away information in situations where humans would not, in 39% and 57% of cases, even when prompts asked for privacy and

agents used chain-of-thought reasoning. Shao et al. (2024) tested models that act as agents with Privacy Lens, which expands privacy-sensitive seeds into full agent trajectories. They found that GPT-4 and

Llama-3-70B leaked information in 25.68 percent and 38.69 percent of cases even when agents were told to protect privacy (see Figure 6).

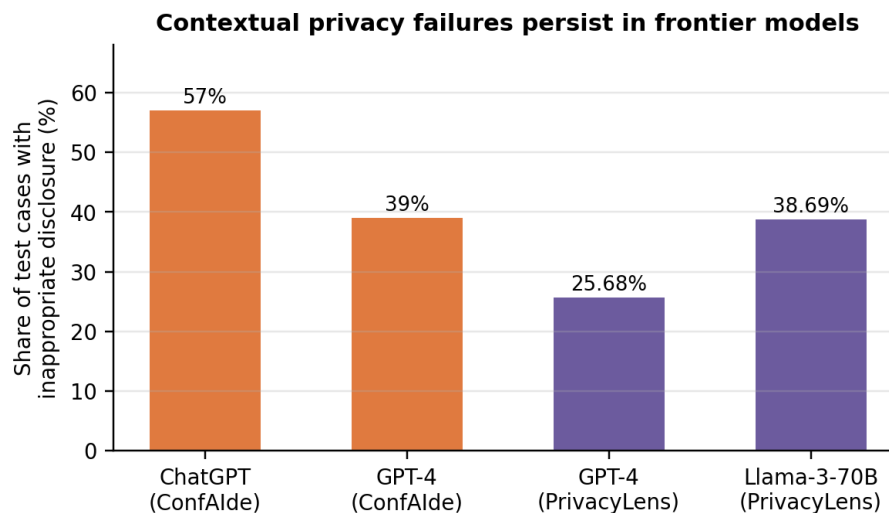


Fig. 6 Rates of inappropriate disclosure reported by two contextual-privacy benchmarks. ConfAlde results (Mireshghallah et al. 2024) and Privacy Lens results with privacy-enhancing instructions (Shao et al. 2024) are not directly comparable, but both show failure rates well above what a sensitive deployment could accept.

A key insight from Privacy Lens is the gap between what models know and how they act. When models answer questions about privacy rules correctly, they often break those rules during real tasks. This shows the issue isn't that the models don't understand what counts as information. They fail to use that knowledge when carrying out multi-step actions. Just giving instructions doesn't fix this problem.

Bagdasarian et al. (2024) found that over-sharing can result from manipulation. In their context-hijacking attack, a third-party app changes the flow of a conversation to trick the model into sharing user data not needed for the task. When tested against a Gemini Ultra agent, this attack lowered

the data-protection rate from 94% to 45% (Fig. 8(B)). This shows how easily privacy can be undermined when models are trained to protect data.

7.4 Inference of personal attributes

The last agent-layer threat involves data that was never stored anywhere. Staab et al. (2024) showed that LLMs can infer attributes such as location, income, age, gender, and more from normal text such as forum posts. On Reddit profiles, the best model achieved 85% top-1 and 95% top-3 accuracy, costing only about one-hundredth of the money and one-two-hundred-fortieth of the time that human annotators need. Standard text anonymization and model alignment did not stop this inference. For agents, the takeaway is that a system that can read a user's messages or documents can create a profile even if none of its data fields are marked as sensitive, and a malicious chatbot could pull out these personal attributes through harmless-seeming conversation

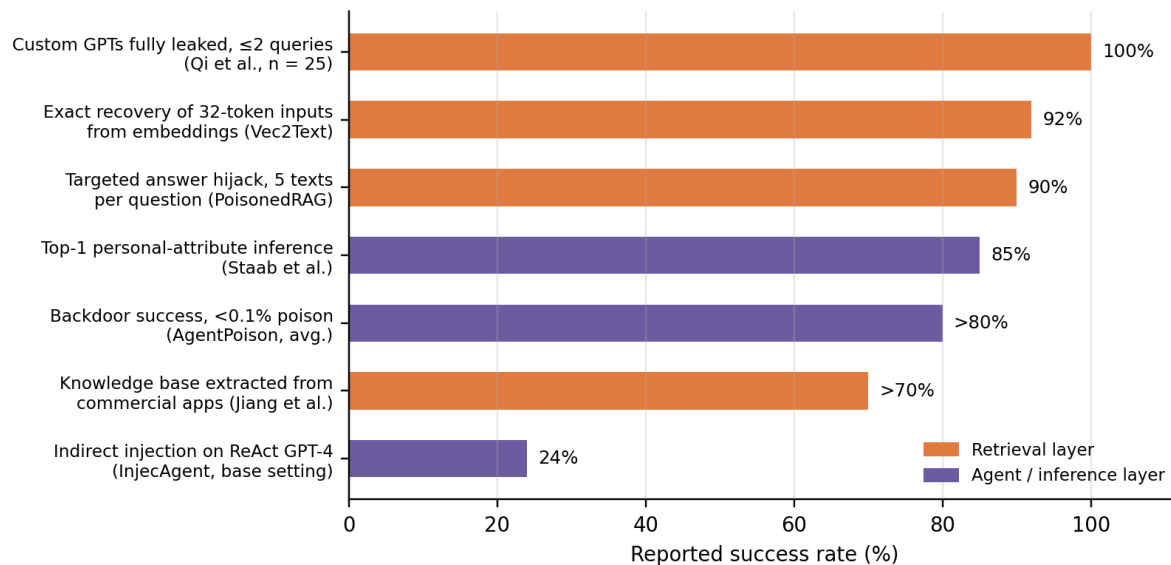


Fig. 7. Success rates reported for representative retrieval-layer and agent-layer attacks. Values come from the original publications and were obtained under different models, datasets, and threat models; they indicate the scale of each risk and should not be read as a ranking.

Figure 7 shows the numbers that were shared in Sections 6 and 7. Two points are worth noting: the numbers are not easy to compare. First, attacks

that target the retrieval layer often achieve success rates above 70% when they have black-box access. The most studied attack at the model layer, which is membership inference on base models, is almost like guessing. This is according to Duan et al. 2024. The real danger has moved from the weights to the system around the model. Second, the smallest number in the chart is 24% for injection against GPT-4. This is still far too high for an agent trusted with an inbox or a bank account

Table 5 summarizes the agent-layer studies.

Table 5: Representative agent-layer (L3) privacy studies

Study	Venue	Threat	Access	Main finding
Greshake et al. (2023)	AISeC@CCS	Indirect injection	prompt Injection	Data theft and self-propagation against real LLM applications
Cohen et al. (2024)	arXiv	Self-replicating worm	Injection	Morris-II spreads between GenAI e-mail assistants while exfiltrating data
Zhan et al. (2024)	Findings ACL	Injection benchmark	Injection	1,054 cases; ReAct GPT-4 compromised in 24% (base)
Debenedetti et al. (2024)	NeurIPS D&B	Injection benchmark	Injection	97 tasks, 629 security cases; extensible harness

Study	Venue	Threat	Access	Main finding
Wang et al. (2025)	ACL	Agent memory extraction	Black-box	MEXTRA extracts other users' memory records
Mireshghallah et al. (2024)	ICLR	Contextual sharing	over-None (benign)	GPT-4 39%, ChatGPT 57% inappropriate disclosure
Shao et al. (2024)	NeurIPS D&B	Over-sharing actions	in None (benign)	GPT-4 25.68%, Llama-3-70B 38.69% with privacy prompts
Bagdasarian et al. (2024)	CCS	Context hijacking	Third-party app	Protection of baseline agent falls from 94% to 45%
Staab et al. (2024)	ICLR	Attribute inference	Black-box	85% top-1 / 95% top-3; anonymization ineffective

8 Defenses

This part covers the defenses, starting with the layers they are meant to protect. Then look at how well they cover those areas and how much they cost. We also try to distinguish between methods that provide a guarantee (a limit that works against any enemy within the model) and those that provide protection tested against known attacks.

8.1 Data-level defenses

The cheapest defense is not to collect or index sensitive data in the first place. Where that is impossible, three data-level techniques are common. *Scrubbing* removes or masks PII with named-entity recognizers before training or indexing; it is easy to deploy but misses context-dependent identifiers and, as Lukas et al. (2023) showed, leaves residual leakage. *Deduplication* removes repeated sequences from training corpora; Kandpal et al. (2022) showed that duplicated sequences are regenerated far more often than unique ones and that deduplication substantially reduces the success of extraction attacks. *Synthetic replacement* substitutes the sensitive corpus with generated text that preserves utility but not identities. Zeng et al. (2024b) applied this idea to retrieval with SAGE, a two-stage method that generates synthetic documents preserving key contextual information and then refines them with agents to strip residual private content; they report utility comparable to the original data with substantially lower leakage.

8.2 Model-level defenses

Differential privacy in training. DP-SGD (Abadi et al. 2016) Is the training-time mechanism that has a formal guarantee. The cost has gone down a lot. Yu et al. (2022) adjusted RoBERTa-Large on MNLI with $\epsilon = 6.7$. Got 87.8% accuracy compared to 90.2% without privacy (Fig. 8(A)); Li et al. (2022) also showed that large, pre-trained models are private learners when adjusted with the right hyperparameters. Both studies found that bigger models lose performance under DP. The biggest recent result is Vault Gemma, a 1-billion-parameter model trained from scratch using sequence-level differential privacy ($\epsilon \leq 2.0$ $\delta \leq 1.1 \times 10^{-10}$). Its performance is like that of private models from about five years ago, such as GPT-2 1.5B. This shows that DP pre-training is possible. It still takes one generation of computing power to achieve this.

Machine unlearning. Unlearning aims to make a trained model behave as if data had never been seen, which is attractive for honoring deletion requests. Early results were encouraging, leading Eldan and Russinovich (2023) to make a Llama-2 model effectively forget the Harry Potter books through fine-tuning on generic substitutes. Benchmarks have since shown how hard reliable unlearning is. On TOFU, a set of 20 fictitious author profiles with 20 question-answer pairs each, none of the baseline methods achieved effective unlearning (Maini et al. 2024). MUSE evaluated eight methods on 7-billion-parameter

models against six criteria and found that most prevented verbatim and knowledge memorization to some degree, but only one avoided severe privacy leakage, and all struggled with utility preservation and sequential deletion requests (Shi et al. 2024b). Unlearning is best understood today as a mitigation, not as a substitute for keeping sensitive data out of training.

Federated and cryptographic training. Federated learning keeps data on client devices and only shares model updates (McMahan et al. 2017); Frameworks like Open Fed LLM bring this approach to instruction tuning of large language models (Ye et al. 2024). Federated learning by itself does not protect against data leakage through gradients or the final model so it is often used with differential privacy or secure aggregation. Homomorphic encryption and secure-party computation enable calculations on encrypted data; THE-X approximated the non-polynomial operations of transformers to make this possible (Chen et al. 2022). These methods protect data but are still too slow for real-time use with large models.

8.3 Retrieval-level defenses

Access control. The most effective retrieval-layer defense is also the most ordinary: filter retrieval results by the permissions of the requesting user, so that no query can retrieve a document its author could not open directly. Many extraction and membership attacks presuppose a shared index accessible to all users; per-user or per-tenant indexing removes that premise. Access control does not, however, protect a user's own documents from an injection that arrives through a different channel.

Differentially private retrieval. Koga et al. (2024) proposed a DP-RAG algorithm that spends privacy budget only on the tokens whose generation requires the sensitive corpus and uses a non-private model for the rest, producing long, accurate answers at $\epsilon \approx 10$. Grislain (2024) proposed a related approach based on private aggregation over retrieved documents. These methods provide formal protection against

extraction and membership inference on the store, although the budgets required for useful answers are still large compared with those used in statistical data release.

Prompt- and output-level controls. Simpler measures include instructions in the RAG template telling the model not to reveal its context, which Anderson et al. (2025) found highly effective for some models and datasets but not others, and post-hoc output filters that block responses with high overlap with retrieved text. Qi et al. (2024) found that eliminating position bias in how retrieved passages are presented substantially reduced extraction. These measures are cheap and worthwhile but are empirical: Li et al. (2024) and Naseh et al. (2025) both designed attacks specifically to evade them.

Embedding protection. Given the success of inversion (Morris et al. 2023), embeddings should be stored with the same access controls and encryption as the source documents. Where embedding must be shared, adding noise or using transformations that degrade inversion is possible, but these trade retrieval qualities for privacy in ways that are still poorly characterized.

8.4 Agent-level defenses

Minimizing what the agent can see. Air Gap Agent (Bagdasarian et al. 2024) restricts the agent's access to only the user data that is relevant to the current task, as judged by a separate minimizer guided by contextual integrity. Under the context-hijacking attack that reduced a baseline Gemini Ultra agent's protection rate to 45%, Air Gap Agent maintained 97% protection (Fig. 8(b)). The principle of data minimization at the level of individual tasks is simple and powerful.

Separating instructions from data. Many defenses aim to stop the model from carrying out instructions that appear in data. Spotlighting marks or encodes input so that the model can recognize where the input came from. On GPT-family models, Spotlighting lowered injection success from over fifty percent to under two percent while leaving task performance unchanged (Hines et al. 2024). StruQ splits the prompt into an instruction

channel and a data channel and then fine-tunes the model to follow the instruction channel. StruQ reports that this method greatly raises resistance to injections with a small loss of usefulness (Chen et al. 2025B). CaMeL takes a stronger, system-level approach: a privileged model turns the trusted user query into an explicit program, a quarantined model processes untrusted data without the ability to change control flow, and capability tags on each value enforce which data may flow to which tool. On Agent Dojo, CaMeL completed 77% of tasks with provable security against the injections

considered, compared with 84% for an undefended system (Debenedetti et al. 2025).

Guardrail models. Classifiers such as Llama Guard (Inan et al. 2023) Check inputs and outputs based on a set of rules. These can be expanded to include categories related to privacy. Cohen et al. (2024) designed a guardrail for their self-replicating worm. It achieved a rate of 1.0 and a false-positive rate of 0.015. Guardrails help as one line of defense. They have the same main problem as all machine-learned detectors. An intelligent attacker can find ways to avoid detection by searching for inputs that the guardrail misses.

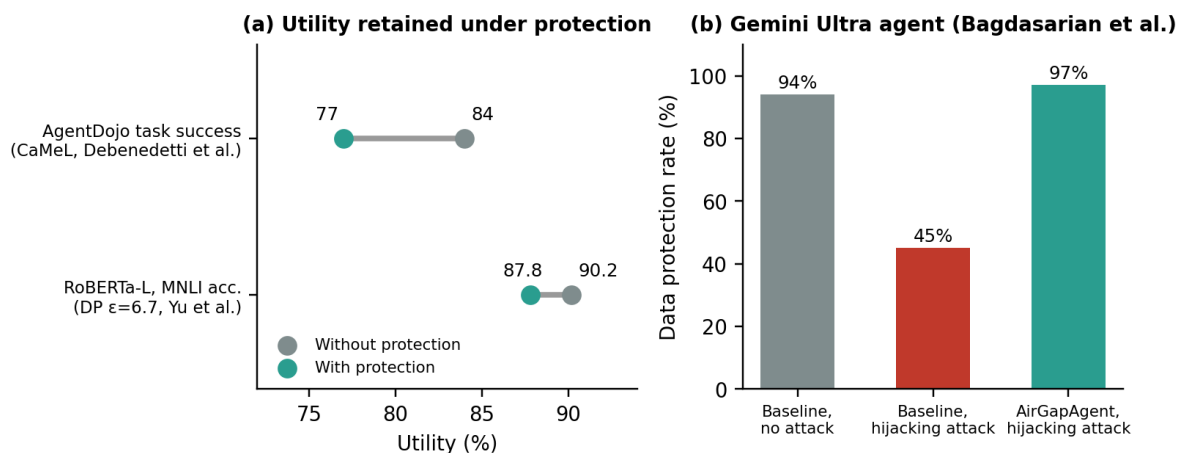


Fig. 8 The cost and benefit of protection, from published results. (a) Utility retained under two defenses: DP fine-tuning (Yu et al. 2022) and the CaMeL agent architecture (Debenedetti et al. 2025). (b) Data-protection rate of a Gemini Ultra agent under a context-hijacking attack, with and without Air Gap Agent (Bagdasarian et al. 2024).

8.5 Hardware-based protection

Trusted execution environments (TEEs) protect data in use by running computation inside hardware-isolated enclaves whose integrity can be attested remotely. Recent data center GPUs support confidential-computing modes that extend this protection to model inference, which allows a provider to process a user's prompt and retrieved documents without the plaintext being visible to the host operating system or cloud operator. TEEs address a threat that none of the other defenses touch: a compromised or curious

infrastructure provider, but they do not prevent the model from leaking data through its outputs. They are therefore complementary to, not a replacement for, the defenses above.

8.6 Coverage and cost

Table 6 summarizes the defense families, and Fig. 9 maps them against the threat classes of Sections 5–7. The map is our qualitative assessment, based on the evidence reviewed; it highlights gaps rather than scoring individual techniques.

Table 6 Defense families: lawyer, guarantee type, strengths and limitations

Defence family	Layer	Guarantee	Strengths	Limitations
PII scrubbing and deduplication	Data	Empirical	Cheap; reduces extraction (Kandpal et al. 2022)	Misses contextual identifiers; residual leakage (Lukas et al. 2023)
Synthetic corpora (e.g. SAGE)	Data / L2	Empirical	Removes identities while keeping utility for RAG	Quality depends on generator; hard to audit
DP training (DP-SGD)	L1	Formal (ϵ , δ)	Bounds per-record leakage; practical for fine-tuning	Utility cost; about one model generation for pre-training
Machine unlearning	L1	Empirical	Supports deletion requests post hoc	Unreliable on TOFU and MUSE; utility loss
Federated cryptographic training	/ L1	Formal (inputs)	Raw data never centralized or visible	Slow for large models; gradients still leak without DP
Access control on retrieval	L2	Deterministic	Removes shared-index premise of most L2 attacks	Does not stop injections via other channels
DP-RAG	L2	Formal (ϵ , δ)	Bounds extraction and MIA on the store	Large budgets ($\epsilon \approx 10$) needed for long answers
Template and output filtering	L2	Empirical	Easy to deploy; blocks naive extraction	Evaded by adaptive attacks (S ² MIA, Riddle)
Task-level data minimization	L3	Empirical	Air Gap Agent keeps 97% protection under hijacking	Relies on a trustworthy minimizer
Instruction/data separation	L3	Empirical / by design	StruQ, spotlighting; CaMeL gives provable flow control	CaMeL costs about 7 points of task success; engineering effort
Guardrail classifiers	L2 / L3	Empirical	Flexible policy enforcement	Adaptive adversaries find gaps
TEEs	Infrastructure	Hardware	Protect data from host and cloud operator	Do not prevent leakage through outputs

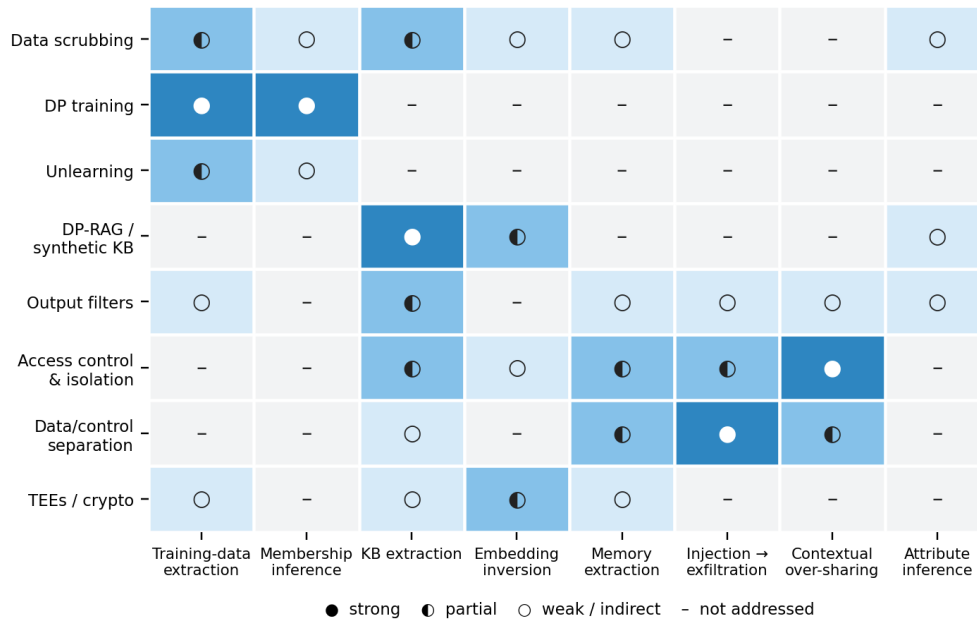


Fig. 9 Coverage of threat classes (columns) by defense families (rows), as assessed in this survey.

Darker cells indicate stronger protection. Model-level defenses cluster on the left; runtime isolation and data/control separation are the only families with strong coverage of agent-layer threats.

Three patterns stand out from the map. First, defenses with guarantees like differential privacy in training and retrieval focus on the model and retrieval layers. They say almost nothing about agent behavior. Second, measures at the agent layer mostly address threats through access isolation and data/control separation. These are strong in practice. They are still new and not

widely used. Third, attribute inference, as discussed in Section 7.4, is poorly addressed by all existing approaches. Since the information is not stored, methods like scrubbing, encryption or unlearning don't work. The only real fix is to limit what the model is allowed to infer and disclose.

Figure 8 shows that the utility cost of protection, where measured, is moderate. For privacy fine-tuning at $\epsilon = 6.7$, the drop in accuracy is 2.4 points. For CaMeL, the task success rate drops by 7 percentage points. Both numbers are acceptable for sensitive applications, and both are trending downward. The bigger hurdle to adoption seems to be engineering complexity, not the loss of accuracy.

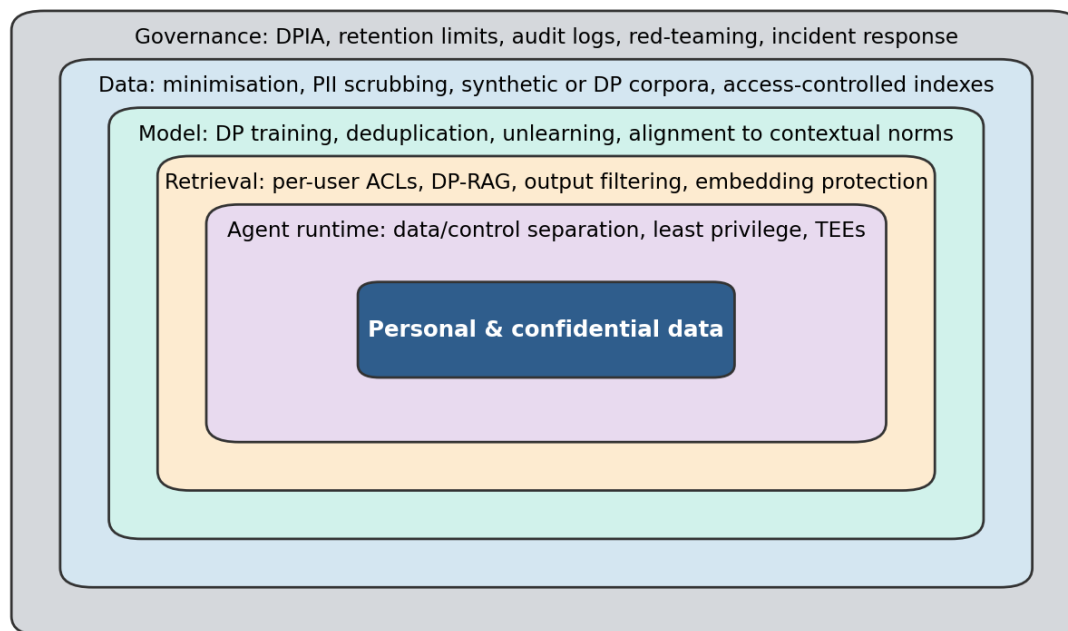


Fig. 10 A defense-in-depth model for compound LLM systems. Each layer addresses a different part of the threat surface; personal data is protected only if every surrounding layer holds.

This leads to the defense-in-depth model in Fig. 10. Because no family covers more than a fraction of the threat surface, a deployment handling sensitive data needs controls at each layer: governance processes that decide what data may be used and for how long; data-level minimization and scrubbing; model-level DP or unlearning where fine-tuning on sensitive data is unavoidable; retrieval-level access control and output filtering; and runtime isolation for agents.

8.7 Two illustrative deployment scenarios

To show how taxonomy and the defense map can be used in practice, we apply them to two hypothetical but typical deployments. Neither scenario describes a real system; both are composites of architectures that are common in current practice. Table 10 summarizes the analysis. *Scenario A: a hospital question-answering assistant.* Clinicians ask questions in language, and the assistant pulls relevant information from discharge

summaries, laboratory reports, and internal guidelines to provide answers. The system uses a large language model that has been lightly fine-tuned on de-identified clinical notes. The biggest risks come from the retrieval layer. A clinician or someone who gains access to a clinician's credentials could use extraction prompts as described by Zeng et al. (2024A). Qi et al. (2024) could extract patient records from the system. They might also use membership inference attacks (Li et al. 2024; Naseh et al. 2025) to determine if a specific person has been treated at the hospital. Because the embedding index contains text vectors, having a copy of the index is essentially like having a copy of the medical records (Morris et al. 2023). The tuned model weights also pose a smaller but lasting risk if the de-identification process missed rare identifiers (Lukas et al. 2023). The analysis highlights a priority order: enforce access control at retrieval time based on the patient and the user's role so a query access only returns records the user is allowed to see. Encrypt the vector store. Treat it as clinical data. Add filtering to prevent outputs from repeating text retrieved from the records. If fine-tuning on

clinical text is kept, use differential privacy fine-tuning, which has a small cost in accuracy at moderate budgets (Yu et al. 2022). DP-RAG (Koga et al. 2024) becomes an option when answering aggregate questions across many patients, especially when users should not see individual records.

Scenario B: a personal e-mail and calendar agent. A consumer agent looks at the user's inbox, writes replies, sets up meetings and remembers things like preferences and contacts over time. In this setup, the biggest risks come from the agent itself. Every email that arrives is content, and it could have hidden instructions inside (Greshake et al. 2023). If an attacker succeeds in injecting these instructions, the agent might forward messages or share calendar details with the sender. Yet self-replicating versions of these attacks can spread to other people's agents (Cohen et al. 2024).

Even without an attacker, the agent might still reveal too much information. For example, it could mention an appointment when suggesting meeting times to a coworker. This kind of over-

sharing has been documented by ConfAIde and Privacy Lens (Mireshghallah et al. 2024; Shao et al. 2024). The agent's memory store ends up holding the kinds of details that MEXTRA-style attacks exploit (Wang et al. 2025).

Looking at the defense map in Figure 9, model-level protections don't do much here. What really helps is how the system is built. Limiting what data the agent sees—showing only what it needs for each task (Bagdasarian et al. 2024) makes a big difference. Separating the user's trusted request from the email content ensures the email cannot change the plan (Debenedetti et al. 2025). Also important: requiring the user to confirm any action that sends data to a person. Giving the user a clear view of what's stored in memory along with an easy way to delete it adds control.

These two situations show something. The same core model can carry different privacy risks depending on the parts around it. The best defenses change accordingly. A privacy review that only checks the model, its training data, its alignment, its fine-tuning would miss the dangers in both cases.

Table 10 Applying the layered analysis to two illustrative deployments

Aspect	A: Hospital QA assistant (RAG)	B: Personal e-mail and calendar agent
Most sensitive assets	Patient records in the retrieval store; embedding index; fine-tuning data	Inbox contents, calendar, contacts, long-term memory
Dominant layer	L2 (retrieval)	L3 (agent)
Most likely attacks	KB extraction, RAG membership inference, embedding theft	Indirect injection via e-mail, contextual over-sharing, memory extraction
Highest-value defenses	Role- and patient-level retrieval ACLs; encrypted vector store; output overlap filter; DP fine-tuning	Task-level data minimization; instruction/data separation; confirmation before outbound data flows; user-visible memory
Residual risk	Authorized insiders; aggregate inference across many answers	Novel injection patterns; subtle over-sharing that passes filters

9 Evaluation practice and benchmarks

The rapid growth of privacy benchmarks is one of the healthier developments in this area. Table 7 lists those most relevant to compound systems.

Table 7 Benchmarks relevant to privacy in compound LLM systems

Benchmark	Source	Layer	What it measures
WikiMIA	Shi et al. (2024a)	L1	Detection of pre-training data by membership signals
MIMIR	Duan et al. (2024)	L1	Membership inference on Pile-trained models across domains and sizes
TOFU	Maini et al. (2024)	L1	Unlearning of 200 fictitious author profiles
MUSE	Shi et al. (2024b)	L1	Six-way evaluation of unlearning, including privacy leakage and sequential requests
ConfAIde	Miresghallah et al. (2024)	L3	Contextual-integrity reasoning in increasingly complex scenarios
Privacy Lens	Shao et al. (2024)	L3	Privacy-norm adherence of agents in full action trajectories
Injec Agent	Zhan et al. (2024)	L2 / L3	Indirect injection with direct-harm and data-theft goals
Agent Dojo	Debenedetti et al. (2024)	L2 / L3	Utility and security of agents under injection; defenses pluggable

Several weaknesses in evaluation practice nevertheless limit what the literature can tell us.

Non-comparable threat models. As Fig. 7 illustrates, headline numbers are rarely comparable. Attacks differ in the access they assume, the size of the corpus, the models tested and the definition of success (verbatim match, ROUGE overlap, attribute accuracy, or task hijack). A shared reporting template—stating adversary access, query budget, corpus size, model version and success criterion would make results far more useful.

Evaluation artefacts. Duan et al. (2024) showed that apparent MIA success can come from temporal distribution shift between member and non-member samples rather than genuine memorization. Similar artefacts are likely in retrieval MIAs whenever members and non-members are drawn from different sources. Benchmarks should construct non-members from the same distribution as members.

Static attacks against adaptive defenses. Most defenses are evaluated against the attacks that existed when they were written. Li et al. (2024) and Naseh et al.

(2025) show how quickly this becomes obsolete. Evaluations should include adaptive attacks designed with knowledge of defense, as is standard practice in adversarial machine learning.

Model churn. Benchmarks report results for specific model versions that are often retired within a year. Results for GPT-4 in 2023 say little about the models deployed in 2026. Open, versioned harnesses such as Agent Dojo that can be re-run on new models are more valuable than one-off measurements.

Missing utility. Many defense papers report privacy gains without a matching utility measurement on realistic tasks. The CaMeL and DP fine-tuning results in Fig. 8 are exceptions that should become the norm.

10 Regulation and governance

We observe how technical controls work within a system that has changed quickly (Fig. 11). Technical controls help keep data safe. The General Data Protection Regulation, also known as GDPR has been in effect in the European Union since May 2018. This rule states principles

of data minimization, purpose limitation, and storage limitation. It also gives people the right to see their data and to have it deleted. The EU Artificial Intelligence Act started in August 2024. Rules for companies that make general-purpose AI models began in August 2025. Outside Europe Californias Consumer Privacy Act (2020) Chinas Personal Information Protection Law (2021) and

Indias Digital Personal Data Protection Act (2023) create rules. The NIST AI Risk Management Framework (NIST 2023) and the OWASP Top 10, for LLM Applications (OWASP 2024) offer advice. The 2025 version of the OWASP list adds information disclosure. It also, for now, includes vector and embedding weaknesses as risks.

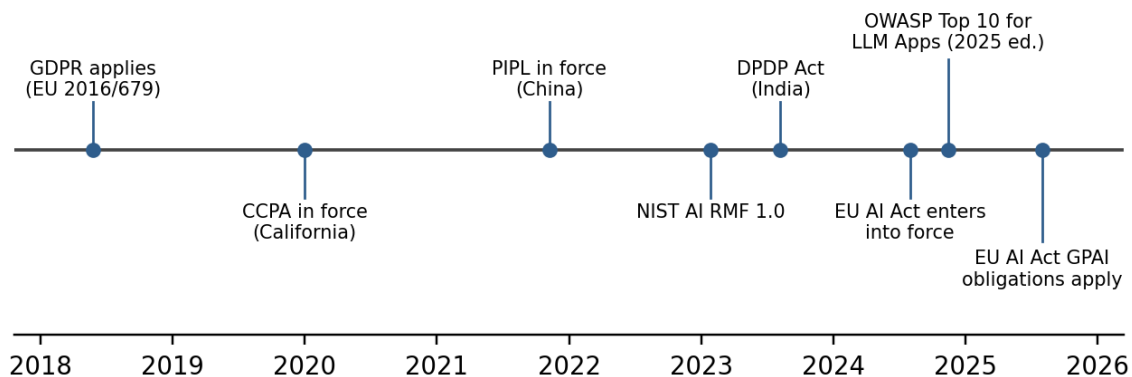


Fig. 11 Timeline of major data-protection laws and AI governance instruments relevant to LLM privacy.

Table 8 connects selected obligations to the technical measures discussed in Section 8. Two tensions stand out. The right to erasure is straightforward to honor for a retrieval store: deleting the document and its embedding, but very difficult for model weights, given the state of unlearning described in Section 8.2. This asymmetry is itself an argument for keeping personal data in retrieval stores rather than in fine-

tuned weights. Second, data-protection impact assessments presuppose that an organization can enumerate where personal data flows; in agentic systems whose tool use is decided at run time, those flows are not fixed in advance. Capability-based architectures such as CaMeL, which make data flows explicit and checkable, are therefore attractive from a compliance perspective as well as a security one.

Table 8: Mapping selected regulatory obligations to technical controls

Obligation	Source	Technical controls	Main difficulty
Data minimization	GDPR Art. 5(1)(c)	Scrubbing, per-task minimization (Air Gap Agent), retention limits on memory	Agents decide data needs at run time
Right to erasure	GDPR Art. 17	Delete from retrieval store and index; unlearning for weights	Unlearning not yet reliable
Security of processing	GDPR Art. 32	Access control, encryption of embeddings, TEEs, injection defenses	Embeddings often treated as non-sensitive

Obligation	Source	Technical controls	Main difficulty
Impact assessment	GDPR Art. 35	Data-flow maps; capability tracking as in CaMeL	Dynamic tool use complicates enumeration
Technical documentation and training-data summary for GPAI	EU AI Act Art. 53	Data provenance records; deduplication and filtering logs	Scale of web corpora
Sensitive information disclosure; vector and embedding weaknesses	OWASP LLM02, LLM08 (2025)	Output filtering, ACL-aware retrieval, embedding protection	Voluntary guidance only

11 Open problems and research directions

The review points to a set of open problems, summarized in Table 9 and discussed below.

Table 9 Open problems and suggested research questions

Open problem	Research question	Promising starting points
Formal privacy for agents	Can contextual norms be expressed as checkable information-flow properties of agent plans?	Capability tagging (CaMeL); contextual integrity
Memory lifecycle	What should an agent remember, for how long, and who may retrieve it?	Retention policies; per-user memory isolation
Multi-agent and protocol privacy	How can provenance and trust be tracked across agents and tool servers?	Signed tool manifests; provenance labels
Inference-time attribute privacy	How can inference of undisclosed attributes be limited without harming reasoning?	Refusal training; output-level attribute filters
Verifiable deletion	Can removal of specific data from weights be certified?	Learning with MIA-based auditing
Whole-system accounting	How do privacy budgets and costs compose across layers?	DP composition; end-to-end red teaming
Multilingual and regional evaluation	How do attacks and PII detectors behave outside English?	Localized benchmarks; regional legal norms

Formal privacy for agents. Differential privacy gives a clean guarantee for training and, increasingly, for retrieval. No comparable notion exists for an agent that reads, reasons, and acts over many steps. Contextual integrity supplies the right intuition, but it has not yet been turned into a verifiable property. A useful first step would be information-flow type systems for agent plans, in the spirit of CaMeL's capability tags, that are

expressive enough to capture contextual norms rather than only confidentiality labels.

Memory lifecycle. Agent memory accumulates indefinitely in most current designs. Principled policies for what is remembered, for how long, and who can retrieve it, together with user-facing controls to inspect and delete memories—remain largely unexplored. MEXTRA (Wang et al. 2025) shows that the absence of such policies is already exploitable.

Multi-agent and protocol-level privacy. As agents increasingly communicate with each other and with third-party tool servers through standard protocols, privacy failures can propagate across organizational boundaries, as the self-replicating worm of Cohen et al. (2024) illustrates. Research is needed on authentication of tool servers, provenance tracking for data passed between agents, and the containment of compromised components.

Inference-time attribute privacy. Staab et al. (2024) show that the most capable models can infer sensitive attributes that were never disclosed. Defenses that limit such inference without crippling general reasoning are almost entirely missing.

Verifiable deletion. Regulators and users need assurance that data has been removed. For retrieval stores, this is an auditing problem; for weights, it remains a research problem. Methods that can certify, even probabilistically, that a model no longer encodes specific data would be of great practical value.

Privacy-utility accounting for whole systems. Most results measure one mechanism in isolation. System designers need methods for composing guarantees and costs across layers—how a DP budget spent on retrieval interacts with a DP budget spent on fine-tuning, or how output filtering changes the effective leakage of an agent.

Low-resource languages and regional contexts. Almost all benchmarks reviewed here are in English, and the PII detectors that underpin scrubbing perform worse on other languages and naming conventions. As LLM services expand in South Asia, Africa and elsewhere, privacy evaluation in those languages and under local legal regimes is urgently needed.

12 Conclusion

The privacy problem in language models has moved. The early focus on what a model memorizes during training remains relevant for fine-tuned models. However, the evidence reviewed in this survey shows that the most practical and successful attacks now target the

components surrounding the model. These components include retrieval stores, embeddings, agent memory and tool channels. Black-box attacks against these components usually succeed. Frontier models that act as agents still overshare information in a quarter or more benchmark cases. Defenses have progressed. Differential privacy is now practical for fine-tuning. It has been shown to work for pre-training and retrieval. Architectural approaches such as task-level data minimization and control/data-flow separation have shown that agents can be made robust at a moderate cost. No single mechanism covers more than a fraction of the threat surface. The risks that arise at the system level contextual over-sharing, attribute inference, cross-agent propagation—remain the least defended. Protecting data in compound large language models will require defense-in-depth evaluation against adaptive adversaries and formal models of privacy, for systems that act as well as speak.

REFERENCES

- Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, Zhang L (2016) Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS), pp 308–318.
<https://doi.org/10.1145/2976749.2978318>
- Anderson M, Amit G, Goldstein A (2025) Is my data in your retrieval database? Membership inference attacks against retrieval-augmented generation. In: Proceedings of the 11th International Conference on Information Systems Security and Privacy (ICISSP). arXiv:2405.20446
- Anthropic (2024) Introducing the Model Context Protocol.
<https://www.anthropic.com/news/model-context-protocol>. Accessed 23 September 2026

- Bagdasarian E, Yi R, Ghalebikesabi S, Kairouz P, Gruteser M, Oh S, Balle B, Ramage D (2024) AirGapAgent: Protecting privacy-conscious conversational agents. In: Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS). arXiv:2405.05175
- Carlini N, Tramèr F, Wallace E, Jagielski M, Herbert-Voss A, Lee K, Roberts A, Brown T, Song D, Erlingsson Ú, Oprea A, Raffel C (2021) Extracting training data from large language models. In: 30th USENIX Security Symposium, pp 2633–2650
- Carlini N, Ippolito D, Jagielski M, Lee K, Tramèr F, Zhang C (2023) Quantifying memorization across neural language models. In: International Conference on Learning Representations (ICLR)
- Chen K, Zhou X, Lin Y, Feng S, Shen L, Wu P (2025a) A survey on privacy risks and protection in large language models. Journal of King Saud University – Computer and Information Sciences 37:163. <https://doi.org/10.1007/s44443-025-00177-1>
- Chen S, Piet J, Sitawarin C, Wagner D (2025b) StruQ: Defending against prompt injection with structured queries. In: 34th USENIX Security Symposium. arXiv:2402.06363
- Chen T, Bao H, Huang S, Dong L, Jiao B, Jiang D, Zhou H, Li J, Wei F (2022) THE-X: Privacy-preserving transformer inference with homomorphic encryption. In: Findings of the Association for Computational Linguistics: ACL 2022
- Chen Z, Xiang Z, Xiao C, Song D, Li B (2024) AgentPoison: Red-teaming LLM agents via poisoning memory or knowledge bases. In: Advances in Neural Information Processing Systems 37 (NeurIPS). arXiv:2407.12784
- Cohen S, Bitton R, Nassi B (2024) Here comes the AI worm: Unleashing zero-click worms that target GenAI-powered applications. arXiv:2403.02817
- Das BC, Amini MH, Wu Y (2025) Security and privacy challenges of large language models: A survey. ACM Computing Surveys 57(6). <https://doi.org/10.1145/3712001>
- DeBenedetti E, Zhang J, Balunović M, Beurer-Kellner L, Fischer M, Tramèr F (2024) AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In: Advances in Neural Information Processing Systems 37, Datasets and Benchmarks Track. arXiv:2406.13352
- DeBenedetti E, Shumailov I, Fan T, Hayes J, Carlini N, Fabian D, Kern C, Shi C, Terzis A, Tramèr F (2025) Defeating prompt injections by design. arXiv:2503.18813
- Duan M, Suri A, Mireshghallah N, Min S, Shi W, Zettlemoyer L, Tsvetkov Y, Choi Y, Evans D, Hajishirzi H (2024) Do membership inference attacks work on large language models? In: Conference on Language Modeling (COLM). arXiv:2402.07841
- Dwork C, Roth A (2014) The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science 9(3–4):211–407
- Eldan R, Russinovich M (2023) Who's Harry Potter? Approximate unlearning in LLMs. arXiv:2310.02238
- European Union (2016) Regulation (EU) 2016/679 of the European Parliament and of the Council (General Data Protection Regulation). Official Journal of the European Union L 119:1–88
- European Union (2024) Regulation (EU) 2024/1689 of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). Official Journal of the European Union, L series, 12 July 2024
- Fu W, Wang H, Gao C, Liu G, Li Y, Jiang T (2024) Membership inference attacks against fine-tuned large language models via self-prompt calibration. In: Advances in Neural Information Processing Systems 37 (NeurIPS)

- Greshake K, Abdelnabi S, Mishra S, Endres C, Holz T, Fritz M (2023) 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), pp 79–90. arXiv:2302.12173
- Grislain N (2024) RAG with differential privacy. arXiv:2412.19291
- He F, Zhu T, Ye D, Liu B, Zhou W, Yu PS (2025) The emerged security and privacy of LLM agent: A survey with case studies. *ACM Computing Surveys* 58(6). <https://doi.org/10.1145/3773080>
- Hines K, Lopez G, Hall M, Zarfati F, Zunger Y, Kiciman E (2024) Defending against indirect prompt injection attacks with spotlighting. arXiv:2403.14720
- Inan H, Upasani K, Chi J, Rungta R, Iyer K, Mao Y, Tontchev M, Hu Q, Fuller B, Testuggine D, Khabisa M (2023) Llama Guard: LLM-based input-output safeguard for human-AI conversations. arXiv:2312.06674
- Jiang C, Pan X, Hong G, Bao C, Chen Y, Yang M (2024) Feedback-guided extraction of knowledge base from retrieval-augmented LLM applications. arXiv:2411.14110
- Kandpal N, Wallace E, Raffel C (2022) Deduplicating training data mitigates privacy risks in language models. In: Proceedings of the 39th International Conference on Machine Learning (ICML), PMLR 162, pp 10697–10707
- Kim S, Yun S, Lee H, Gubri M, Yoon S, Oh SJ (2023) ProPILE: Probing privacy leakage in large language models. In: Advances in Neural Information Processing Systems 36 (NeurIPS)
- Koga T, Wu R, Zhang Z, Chaudhuri K (2024) Privacy-preserving retrieval-augmented generation with differential privacy. arXiv:2412.04697
- Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Küttler H, Lewis M, Yih W, Rocktäschel T, Riedel S, Kiela D (2020) Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Advances in Neural Information Processing Systems 33 (NeurIPS), pp 9459–9474
- Li X, Tramèr F, Liang P, Hashimoto T (2022) Large language models can be strong differentially private learners. In: International Conference on Learning Representations (ICLR)
- Li Y, Liu G, Wang C, Yang Y (2024) Generating is believing: Membership inference attacks against retrieval-augmented generation. arXiv:2406.19234
- Liu M, Zhang S, Long C (2025) Mask-based membership inference attacks for retrieval-augmented generation. In: Proceedings of the ACM Web Conference 2025 (WWW). <https://doi.org/10.1145/3696410.3714771>
- Lukas N, Salem A, Sim R, Tople S, Wutschitz L, Zanella-Béguelin S (2023) Analyzing leakage of personally identifiable information in language models. In: IEEE Symposium on Security and Privacy (S&P), pp 346–363
- Maini P, Feng Z, Schwarzschild A, Lipton ZC, Kolter JZ (2024) TOFU: A task of fictitious unlearning for LLMs. arXiv:2401.06121
- McMahan B, Moore E, Ramage D, Hampson S, Agüera y Arcas B (2017) Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), PMLR 54, pp 1273–1282
- Mireshghallah N, Kim H, Zhou X, Tsvetkov Y, Sap M, Shokri R, Choi Y (2024) Can LLMs keep a secret? Testing privacy implications of language models via contextual integrity theory. In: International Conference on Learning Representations (ICLR). arXiv:2310.17884

- Morris JX, Kuleshov V, Shmatikov V, Rush AM (2023) Text embeddings reveal (almost) as much as text. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)
- Morris JX, Zhao W, Chiu JT, Shmatikov V, Rush AM (2024) Language model inversion. In: International Conference on Learning Representations (ICLR). arXiv:2311.13647
- Naseh A, Peng Y, Suri A, Chaudhari H, Oprea A, Houmansadr A (2025) Riddle me this! Stealthy membership inference for retrieval-augmented generation. In: Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS).
<https://doi.org/10.1145/3719027.3744840>
- Nasr M, Carlini N, Hayase J, Jagielski M, Cooper AF, Ippolito D, Choquette-Choo CA, Wallace E, Tramèr F, Lee K (2023) Scalable extraction of training data from (production) language models. arXiv:2311.17035
- Neel S, Chang P (2023) Privacy issues in large language models: A survey. arXiv:2312.06717
- Nissenbaum H (2004) Privacy as contextual integrity. *Washington Law Review* 79(1):119
- NIST (2023) Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1. National Institute of Standards and Technology.
<https://doi.org/10.6028/NIST.AI.100-1>
- OWASP Foundation (2024) OWASP Top 10 for LLM Applications 2025. OWASP GenAI Security Project.
<https://genai.owasp.org/llm-top-10/>. Accessed 23 September 2026
- Qi Z, Zhang H, Xing E, Kakade S, Lakkaraju H (2024) Follow my instruction and spill the beans: Scalable data extraction from retrieval-augmented generation systems. arXiv:2402.17840
- Shao Y, Li T, Shi W, Liu Y, Yang D (2024) PrivacyLens: Evaluating privacy norm awareness of language models in action. In: Advances in Neural Information Processing Systems 37, Datasets and Benchmarks Track. arXiv:2409.00138
- Shi W, Ajith A, Xia M, Huang Y, Liu D, Blevins T, Chen D, Zettlemoyer L (2024a) Detecting pretraining data from large language models. In: International Conference on Learning Representations (ICLR). arXiv:2310.16789
- Shi W, Lee J, Huang Y, Malladi S, Zhao J, Holtzman A, Liu D, Zettlemoyer L, Smith NA, Zhang C (2024b) MUSE: Machine unlearning six-way evaluation for language models. arXiv:2407.06460
- Shokri R, Stronati M, Song C, Shmatikov V (2017) Membership inference attacks against machine learning models. In: IEEE Symposium on Security and Privacy (S&P), pp 3–18
- Staab R, Vero M, Balunović M, Vechev M (2024) Beyond memorization: Violating privacy via inference with large language models. In: International Conference on Learning Representations (ICLR). arXiv:2310.07298
- VaultGemma Team (2025) VaultGemma: A differentially private Gemma model. Technical report, Google Research and Google DeepMind. arXiv:2510.15001
- Wang B, He W, Zeng S, Xiang Z, Xing Y, Tang J, He P (2025) Unveiling privacy risks in LLM agent memory. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL). arXiv:2502.13172
- Yan B, Li K, Xu M, Dong Y, Zhang Y, Ren Z, Cheng X (2024) On protecting the data privacy of large language models (LLMs): A survey. arXiv:2403.05156
- Yao S, Zhao J, Yu D, Du N, Shafran I, Narasimhan K, Cao Y (2023) ReAct: Synergizing reasoning and acting in language models. In: International Conference on Learning Representations (ICLR)

Yao Y, Duan J, Xu K, Cai Y, Sun Z, Zhang Y (2024)
A survey on large language model (LLM)
security and privacy: The good, the bad, and
the ugly. *High-Confidence Computing*
4(2):100211

Ye R, Wang W, Chai J, Li D, Li Z, Xu Y, Du Y,
Wang Y, Chen S (2024) OpenFedLLM:
Training large language models on
decentralized private data via federated
learning. In: *Proceedings of the 30th ACM
SIGKDD Conference on Knowledge
Discovery and Data Mining (KDD)*

Yu D, Naik S, Backurs A, Gopi S, Inan HA,
Kamath G, Kulkarni J, Lee YT, Manoel A,
Wutschitz L, Yekhanin S, Zhang H (2022)
Differentially private fine-tuning of
language models. In: *International
Conference on Learning Representations
(ICLR)*. arXiv:2110.06500

Zeng S, Zhang J, He P, Xing Y, Liu Y, Xu H, Ren
J, Wang S, Yin D, Chang Y, Tang J (2024a)
The good and the bad: Exploring privacy
issues in retrieval-augmented generation
(RAG). In: *Findings of the Association for
Computational Linguistics: ACL 2024*.
arXiv:2402.16893

Zeng S, Zhang J, He P, Ren J, Zheng T, Lu H, Xu
H, Liu H, Xing Y, Tang J (2024b) Mitigating
the privacy issues in retrieval-augmented
generation (RAG) via pure synthetic data.
arXiv:2406.14773

Zhan Q, Liang Z, Ying Z, Kang D (2024)
InjecAgent: Benchmarking indirect prompt
injections in tool-integrated large language
model agents. In: *Findings of the
Association for Computational Linguistics:
ACL 2024*. arXiv:2403.02691

Zhang Y, Carlini N, Ippolito D (2024) Effective
prompt extraction from language models.
arXiv:2307.06865

Zou W, Geng R, Wang B, Jia J (2025)
PoisonedRAG: Knowledge corruption
attacks to retrieval-augmented generation of
large language models. In: *34th USENIX
Security Symposium*. arXiv:2402.07867