On Private Knowledge Retrieval and Private Inference for secure multi-agent systems

V. Torra[1], M. Bras-Amorós[2]

[1] Umeå University, Sweden
[2] Universitat Politècnica de Catalunya

September 15, 2026

Multi-agent AI

- Agentic AI, agents running Large Language Models (LLMs)
- Agents: autonomous, access to other software agents, make decisions, ...

AI Security: Memory poisoning attacks

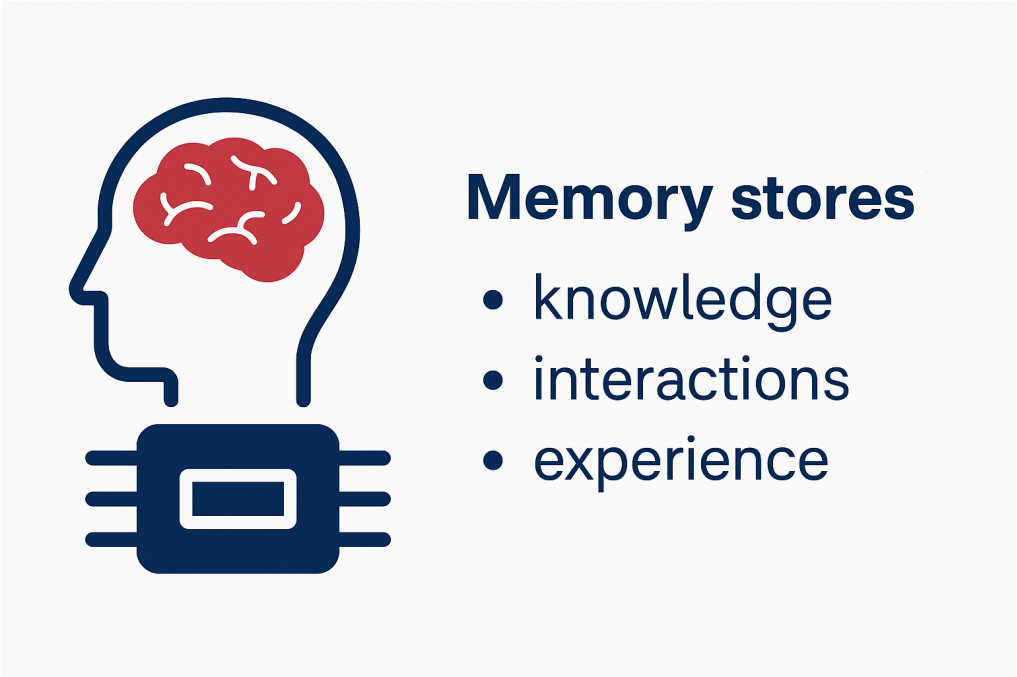
- AI Security challenges for Agentic AI/MAS
  - Multiagent systems live in complex, adversarial environments, including malicious agents
  - Agentic AI/MAS are not one-shot applications: value emerges through ongoing interactions over time

AI Security: Memory poisoning attacks

- AI Security challenges for Agentic AI/MAS
  - Multiagent systems live in complex, adversarial environments, including malicious agents
  - Agentic AI/MAS are not one-shot applications: value emerges through ongoing interactions over time
- Memory is a core component of agents behavior
- Memory poisoning attacks pose a serious threat to safety

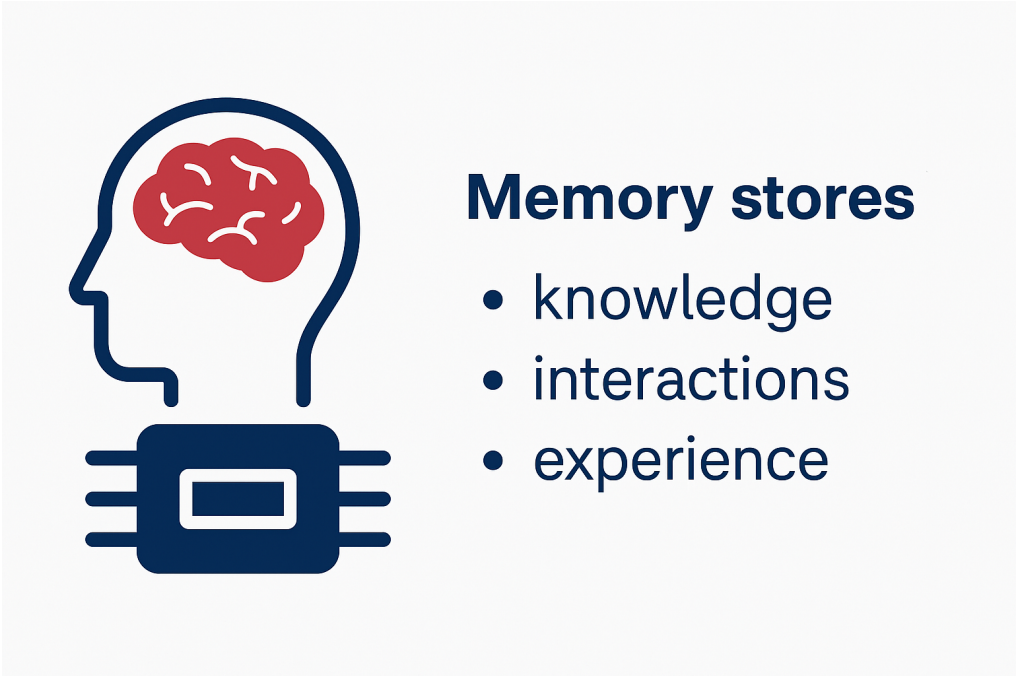
Memory systems for LLM-multi-agent AI

- Memory, not only data



Memory systems for LLM-multi-agent AI

- Memory, not only data

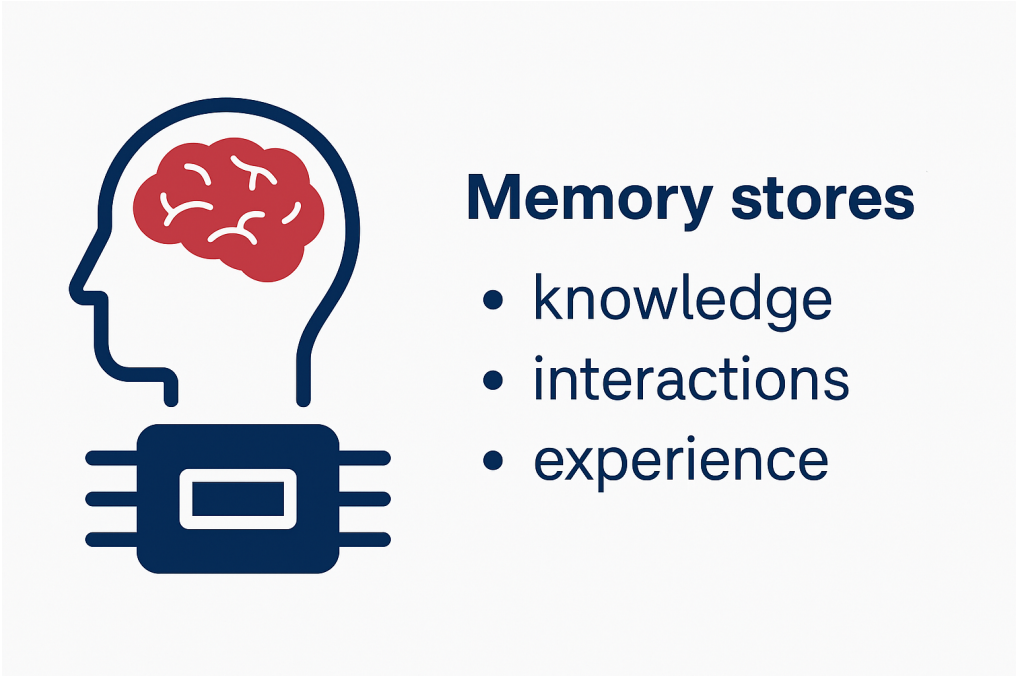


It may be used to

- make inferences,

Memory systems for LLM-multi-agent AI

- Memory, not only data

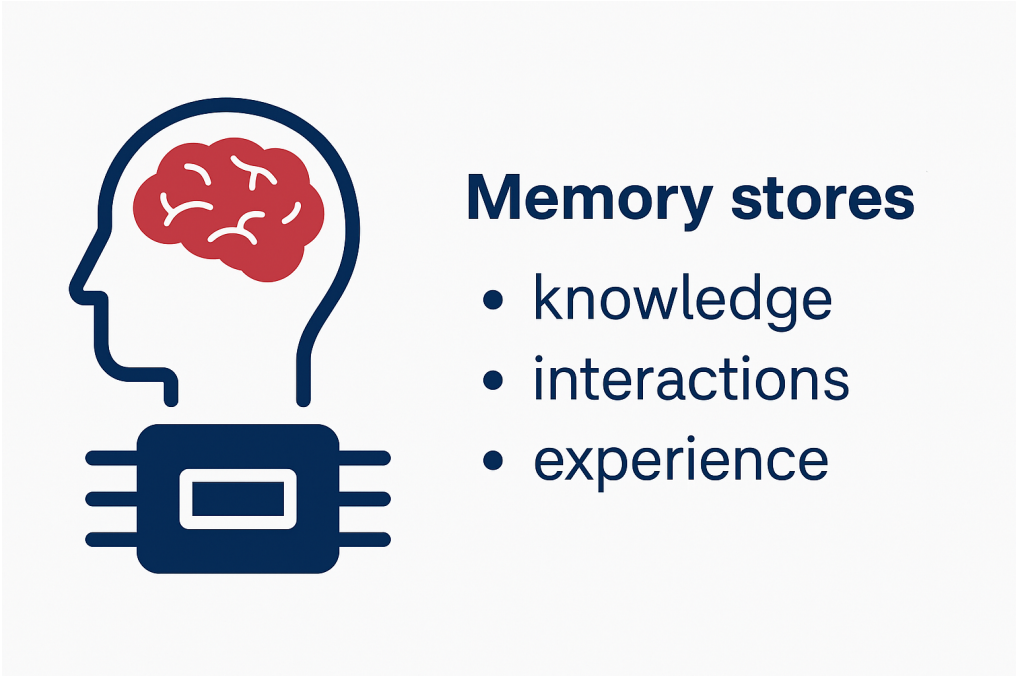


It may be used to

- make inferences,
- make decisions,

Memory systems for LLM-multi-agent AI

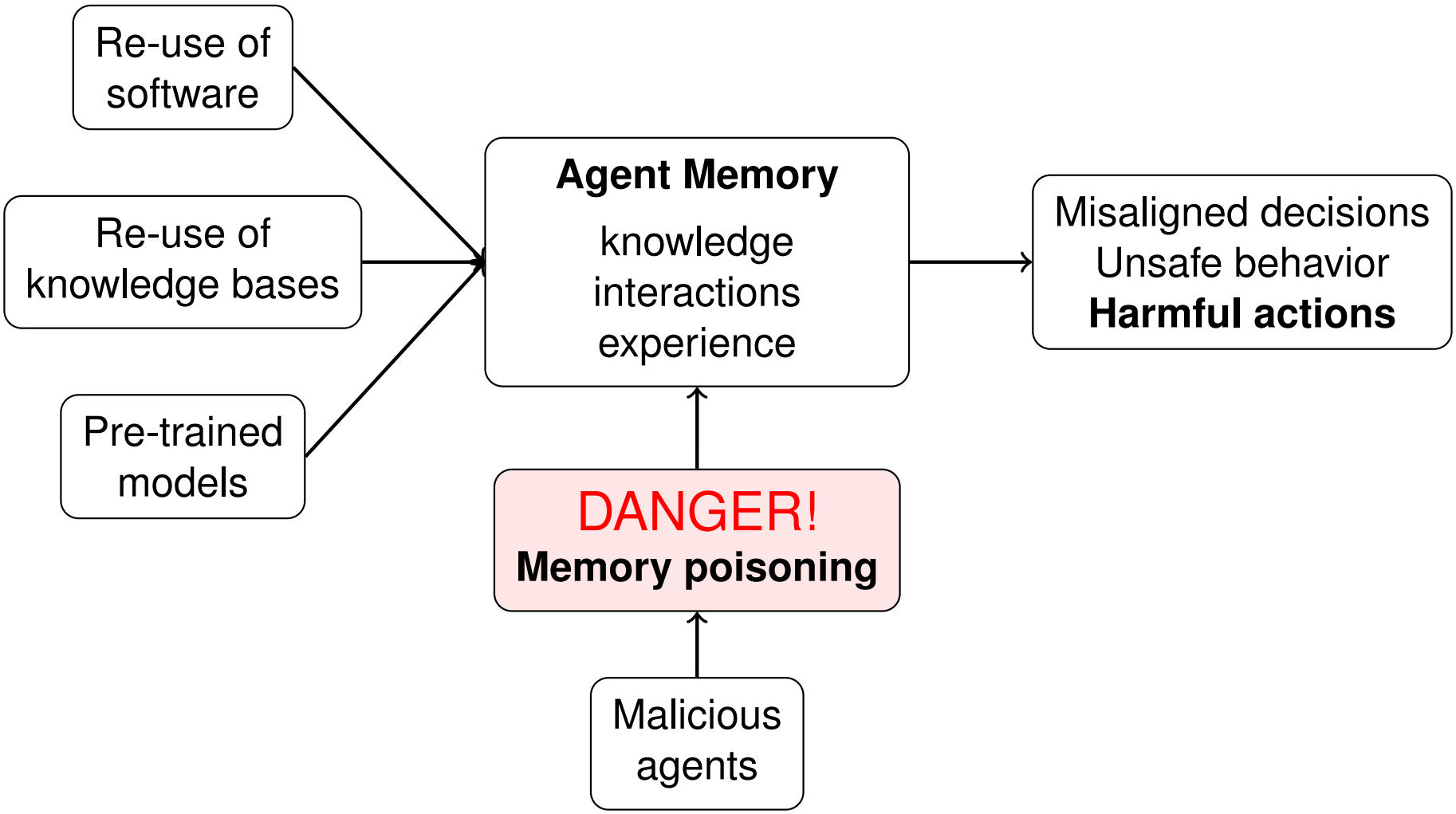
- Memory, not only data



It may be used to

- make inferences,
- make decisions,
- act.

Memory poisoning



Different memory systems for LLM-multi-agent AI

Semantic memory

Episodic memory

Short-term memory

Different memory systems for LLM-multi-agent AI

Semantic memory

Episodic memory

Short-term memory

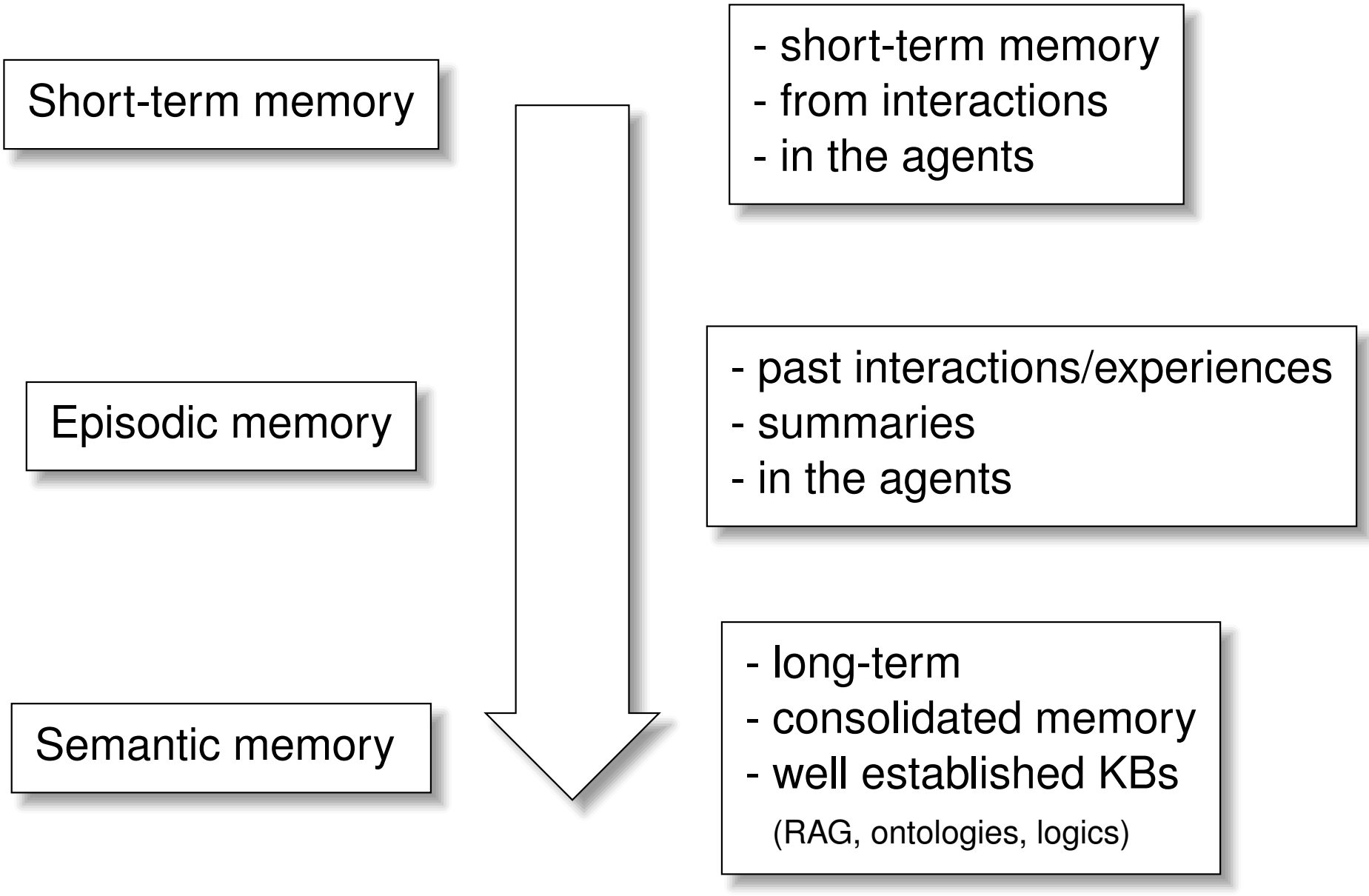
The different systems differ in

duration

origin/source

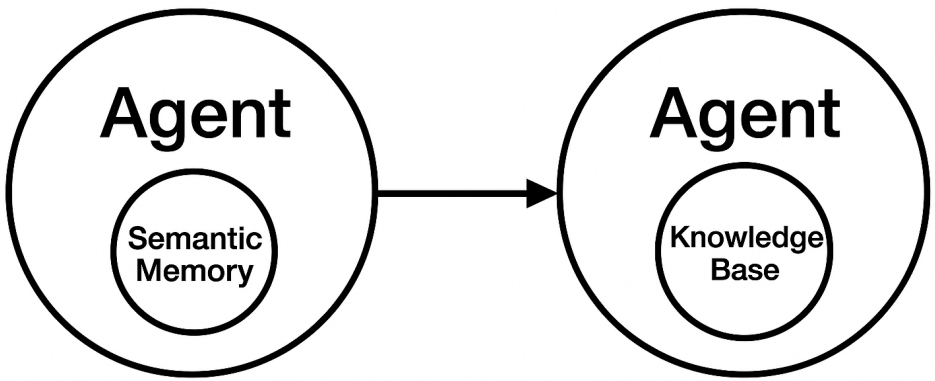
localization

Memory systems for LLM-multi-agent AI



Semantic memory attacks

- Attacks (changes / updates) to internal memory
- Attacks (changes / updates) to external memory
- Attacks in communications

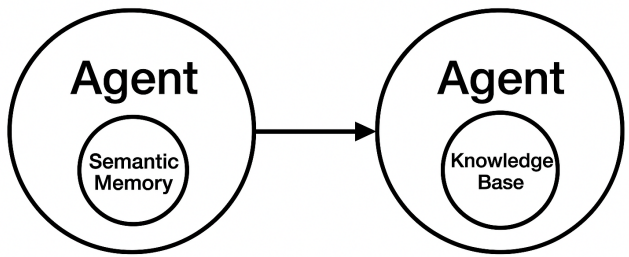


Semantic memory mitigation strategies

- (ia) Secure memory mechanisms
  - Hash + signatures to track/detect undesired changes
  - Internal memory: Cryptography to make $\mathcal{D}_{clean}$ unknown to attackers

Semantic memory mitigation strategies

- (ib) Secure memory mechanisms
  and Secure inference in external memory
  - Fully Homomorphic Encryption (FHE)
    Solutions for e.g. decision trees, for small neural networks, costly for larger models
  - External knowledge bases but local inference
    based on private information retrieval (logic-based inference)



Semantic memory mitigation strategies

- (ic) Secure memory mechanisms and Secure inference in external memory
  - Internal inference system (local to the agent)
  - External semantic memory

Semantic memory mitigation strategies

- (ii) Secure updating of knowledge bases
  - Secure provenance structures to ensure integrity / non-forgeability of the knowledge base
    Who supplied knowledge? who updated knowledge?
  - Provenance itself should be secure
  - knowledge items include basic provenance

E1: $\forall$ x, man(x) $\Rightarrow$ mortal(x) [A]
E2: man(socrates) [A]

E3: $\forall$ x, woman(x) $\Rightarrow$ mortal(x) [B]
E4: woman(hypathia) [B]

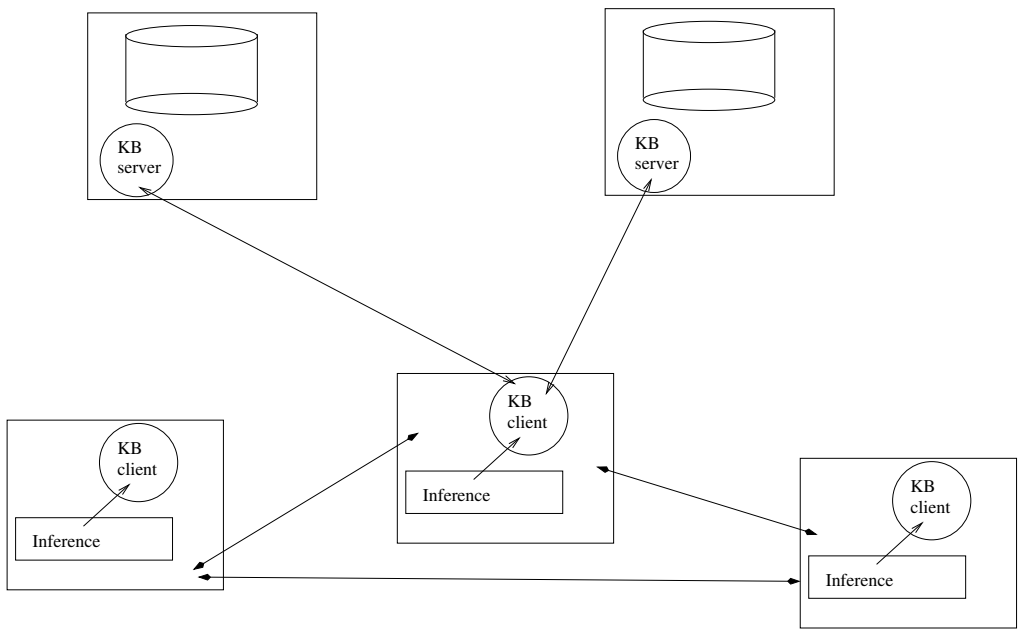
E5: $\forall$ x, car(x) $\Rightarrow$ vehicle(x) [C]
E6: $\forall$ x, bicycle(x) $\Rightarrow$ vehicle(x) [C]

Semantic memory mitigation strategies

- (iii) Secure communications
  - Communication with (trusted) agents should be secured
  - We do not implement secure communication, our system runs in a single machine.

Private inference in `tipp++`

- Private knowledge retrieval and private inference
  - Agents provide knowledge as a service
  - Local inference mechanisms (Prolog-like)



Private inference in `tipp++`

- Prolog-like inference: local inference engine
  - Starts with a list of goals (e.g., `?- mortal(hypathia)`)
  - Repeat
  - Find a predicate that unifies/matches
  - Replace the goal by subgoals (if any)
  - If the goal list is empty, predicate is proven
  - Otherwise, solve the goals (recursion)
- Standard semantics: closed-world assumption (CWA)
  (a non-provable query is false)
- Example:
  `?- mortal(hypathia)` ⇒ `man(hypathia)` ⇒ `fail`
  backtracking
  `?- mortal(hypathia)` ⇒ `woman(hypathia)` ⇒ `T`

Private inference in `tipp++`

- Knowledge-base client (KB client)
  - Local inference engine queries KB client using pair (predicate name, arity)

Private inference in `tipp++`

- Knowledge-base client (KB client)
  - Local inference engine queries KB client using pair (predicate name, arity)
  - If the KB client has this knowledge, returns it
  - If the KB client does not have this knowledge:
    - prepares an external privacy-preserving query
    - accesses to external query through hash/index functions

Private inference in `tipp++`

- Knowledge-base client (KB client)
  - Local inference engine queries KB client using pair (predicate name, arity)
  - If the KB client has this knowledge, returns it
  - If the KB client does not have this knowledge:
    - prepares an external privacy-preserving query
    - accesses to external query through hash/index functions
  - The agent verifies integrity using associated provenance information and digital signature. If verification successful:

Private inference in `tipp++`

- Knowledge-base client (KB client)
  - Local inference engine queries KB client using pair (predicate name, arity)
  - If the KB client has this knowledge, returns it
  - If the KB client does not have this knowledge:
    - prepares an external privacy-preserving query
    - accesses to external query through hash/index functions
  - The agent verifies integrity using associated provenance information and digital signature. If verification successful:
  - Knowledge returned from the external query is cached because (typically)
    - rules are recursive, and
    - solving a predicate requires multiple access to the same facts.

Private inference in `tipp++`

- Example:
  - query `mortal(hypathia)`,
  - forces the local KB-client to query the external knowledge base server for:
    `(mortal,1)`
  - and will retrieve (and catch) knowledge with index $i = hash((mortal, 1))$ the two rules seen before:
    $\forall$ x, man(x) $\Rightarrow$ mortal(x)
    $\forall$ x, woman(x) $\Rightarrow$ mortal(x)
  - Their internal representation are:

```
("mortal", 1): [
    {"head": ["mortal","X"],
     "body": [["man","X"]],
     "meta": {"topic":"socrates"},"source":"A"},
    {"head": ["mortal","X"],
     "body": [["woman","X"]],
     "meta": {"topic":"socrates"},"source":"B"}, ],
```

Private inference in `tipp++`

- Knowledge-base server and private knowledge retrieval
  - Single server, k-anonymity. The real index i and $k-1$ other random indices.
  - Single server, b-buckets. Knowledge base is partitioned into buckets, and the bucket with all indices modulo b equal to i are retrieved.
  - Two non-colluding knowledge bases.
    - Access through Private Information Retrieval using $i = hash(name, arity)$.
    - The KB client builds two random queries differing in the bit value associated to i
      q_1 = 0100101*0*10010101001
      q_2 = 0100101*1*10010101001
    - Knowledge base is padded

Private Information Retrieval (PIR)

- How a user can retrieve an object from a database (or a search engine) in such a way that the database server is not aware of the object being retrieved.
- Definition. Let the database be represented by a binary string of n bits $x = x_0, \ldots, x_{n-1}$. Therefore, $x_i \in \{0, 1\}$. Given a value $i \in \{0, \ldots, n-1\}$, the private information retrieval problem consists of retrieving x_i without the database server knowing which is the bit being retrieved.

(Information Theoretic) Private Information Retrieval

- Information theoretic (IT-PIR): cannot be broken with unlimited computing power

(Information Theoretic) Private Information Retrieval

- Information theoretic (IT-PIR): cannot be broken with unlimited computing power
- Theorem. Every (information theoretic) PIR scheme with a single-database (with n bits) requires $\Omega(n)$ bits of communication.

(Information Theoretic) Private Information Retrieval

- Information theoretic (IT-PIR): cannot be broken with unlimited computing power
- Theorem. Every (information theoretic) PIR scheme with a single-database (with n bits) requires $\Omega(n)$ bits of communication.
- It can be proven (Chor et al. 1998) that
  if a user wants to keep its privacy (in the information theoretic sense), then essentially the only thing she/he can do is to ask for a copy of the whole database.

IT-Private Information Retrieval

- Communication complexity is reduced: sublinear in n by assuming that the data is replicated.

IT-Private Information Retrieval

- Communication complexity is reduced: sublinear in n by assuming that the data is replicated.
  - k copies of the database are considered
  - DB copies do not collaborate

IT-Private Information Retrieval

- Communication complexity is reduced: sublinear in n by assuming that the data is replicated.
  - k copies of the database are considered
  - DB copies do not collaborate
- **Example.** Scheme in (Chor et al., 1999) with communication complexity $O(n^{1/3})$ for $k = 2$

IT-Private Information Retrieval: k copies

- Problem. (DB not being intercommunicated)
  - Database. A binary string $x = x_0 \cdots x_{n-1}$ of length n (Identical copies of this string are stored in $k \geq 2$ servers)
  - User. Given index i, is interested in obtaining the value of bit x_i
  - *Solution:* The user queries each of the servers and gets replies from which the desired bit x_i can be computed. The server does not gain any information about i from the query.

IT-Private Information Retrieval: Problem Definition

- Input
  - $i \in [n]$ where $[n] = \{0, \ldots, n-1\}$
  - r random input of length ℓ_{rnd}
- Overview of the process
  - k queries $Q_1(i, r), \ldots, Q_k(i, r)$ of length ℓ_q each
  - Servers respond according to strategies $A_1, \ldots, A_k$ with replies of length ℓ_a according to the content of the DB x
  - The user reconstructs the desired bit x_i from the k replies, together with i and r

IT-Private Information Retrieval: Formalization

- A k-server PIR scheme for database length n consists of
  - k query functions $Q_1, \ldots, Q_k : [n] \times \{0,1\}^{\ell_{rnd}} \to \{0,1\}^{l_q}$
  - k answer functions, $A_1, \ldots, A_k : \{0,1\}^n \times \{0,1\}^{l_q} \to \{0,1\}^{l_a}$
  - a reconstruction function
    $R : [n] \times \{0,1\}^{l_{rnd}} \times (\{0,1\}^{l_a})^k \to \{0,1\}$
- These functions should satisfy

IT-Private Information Retrieval: Formalization

- A k-server PIR scheme for database length n consists of
  - k query functions $Q_1, \ldots, Q_k : [n] \times \{0,1\}^{\ell_{rnd}} \to \{0,1\}^{l_q}$
  - k answer functions, $A_1, \ldots, A_k : \{0,1\}^n \times \{0,1\}^{l_q} \to \{0,1\}^{l_a}$
  - a reconstruction function $R : [n] \times \{0,1\}^{l_{rnd}} \times (\{0,1\}^{l_a})^k \to \{0,1\}$
- These functions should satisfy
  - Correctness. For every $x \in \{0,1\}^n$, $i \in [n]$, and $r \in \{0,1\}^{\ell_{rnd}}$

$$R(i, r, A_1(x, Q_1(i, r)), \ldots, A_k(x, Q_k(i, r))) = x_i$$

IT-Private Information Retrieval: Formalization

- A k-server PIR scheme for database length n consists of
  - k query functions $Q_1, \ldots, Q_k : [n] \times \{0,1\}^{\ell_{rnd}} \to \{0,1\}^{l_q}$
  - k answer functions, $A_1, \ldots, A_k : \{0,1\}^n \times \{0,1\}^{l_q} \to \{0,1\}^{l_a}$
  - a reconstruction function $R : [n] \times \{0,1\}^{l_{rnd}} \times (\{0,1\}^{l_a})^k \to \{0,1\}$
- These functions should satisfy
  - Correctness. For every $x \in \{0,1\}^n$, $i \in [n]$, and $r \in \{0,1\}^{\ell_{rnd}}$
    $$R(i, r, A_1(x, Q_1(i,r)), \ldots, A_k(x, Q_k(i,r))) = x_i$$
  - Privacy. For every $i, j \in [n]$, $s \in [k]$, and $q \in \{0,1\}^{l_q}$
    $$Pr(Q_s(i,r) = q) = Pr(Q_s(j,r) = q)$$
    where the probabilities are taken over uniformly chosen $r \in \{0,1\}^{\ell_{rnd}}$

IT-PIR: case k=2 databases

- Notation.
  - $k = 2$, databases: SDB_1, SDB_2
  - index of the bit of interest: $i \in \{0, \ldots, n-1\}$
- We will use: For $S \subset \{0, \ldots, n-1\}$, and i as above, $S \boxplus i$ is defined as

$$S \boxplus i = \begin{cases} S \cup \{i\} & \text{if } i \notin S \\ S \setminus \{i\} & \text{if } i \in S \end{cases} \tag{1}$$

IT-PIR: case k=2 databases: Scheme

- **Step 1.** User selects $S \subset \{0, \ldots, n-1\}$, index j with prob. 1/2

IT-PIR: case k=2 databases: Scheme

- **Step 1.** User selects $S \subset \{0, \ldots, n-1\}$, index j with prob. 1/2
- **Step 2.** User sends $Q_1 = S$ to server SDB_1, $Q_2 = S \boxplus i$ to SDB_2.
  Here, $Q_1 = b_0^1 \ldots b_{n-1}^1$ where $b_i^1 = 1$ if and only if $i \in S$ and $Q_2 = b_0^2 \ldots b_{n-1}^2$ where $b_i^2 = 1$ if and only if $i \in S \boxplus i$.

IT-PIR: case k=2 databases: Scheme

- **Step 1.** User selects $S \subset \{0, \ldots, n-1\}$, index j with prob. 1/2
- **Step 2.** User sends $Q_1 = S$ to server SDB_1, $Q_2 = S \boxplus i$ to SDB_2.
  Here, $Q_1 = b_0^1 \ldots b_{n-1}^1$ where $b_i^1 = 1$ if and only if $i \in S$ and $Q_2 = b_0^2 \ldots b_{n-1}^2$ where $b_i^2 = 1$ if and only if $i \in S \boxplus i$.
- **Step 3.** Server SDB_1 sends an exclusive-or of the bits in S, server SDB_2 sends an exclusive or of the bits in $S \boxplus i$.

$$A_1 = \bigoplus_{j \in S} x_j$$

$$A_2 = \bigoplus_{j \in S \boxplus i} x_j.$$

IT-PIR: case k=2 databases: Scheme

- **Step 1.** User selects $S \subset \{0, \ldots, n-1\}$, index j with prob. 1/2
- **Step 2.** User sends $Q_1 = S$ to server SDB_1, $Q_2 = S \boxplus i$ to SDB_2.
  Here, $Q_1 = b_0^1 \ldots b_{n-1}^1$ where $b_i^1 = 1$ if and only if $i \in S$ and $Q_2 = b_0^2 \ldots b_{n-1}^2$ where $b_i^2 = 1$ if and only if $i \in S \boxplus i$.
- **Step 3.** Server SDB_1 sends an exclusive-or of the bits in S, server SDB_2 sends an exclusive or of the bits in $S \boxplus i$.

$$\begin{aligned} A_1 &= \bigoplus_{j \in S} x_j \\ A_2 &= \bigoplus_{j \in S \boxplus i} x_j. \end{aligned}$$

- **Step 4.** User computes exclusive or of the bits received.

$$R = A_1 \oplus A_2.$$

IT-PIR: case k=2 databases

- Correctness of the procedure. R results into x_i.
  - We use the property that the xor of x_j with x_j is always zero: i.e., $x_j \oplus x_j = 0$
  - Then,

$$R = A_1 \oplus A_2 = \bigoplus_{j \in S} x_j \bigoplus_{j \in S \boxplus i} x_j = \bigoplus_{j \in S \setminus \{i\}} (x_j \oplus x_j) \oplus x_i = x_i.$$

- Privacy.
  - The servers do not get any information about the index desired by the user. I.e., the privacy condition above is satisfied. This is so because each server receives a uniformly distributed subset of $\{0, \ldots, n-1\}$.
- Communication.
  - The solution requires the servers to send a single bit (i.e., $l_a = 1$). Note that the number of bits sent by the user has length n.

Private inference in `tipp++`

- Complexity analysis

	knowledge bases	sent communication	received communication	selection/ computation
k-anonymity	1 KB	$k \cdot 64$	$k \cdot r$	select 1 of k
b-buckets	1 KB	$1 \cdot 64$	$m \cdot r$	select 1 of b
2-PIR	2 KB	$2 \cdot n_{pa}$	$2 \cdot r$	XOR($2r$)
2^d-PIR	d KB	$d \cdot n_{pa}^{1/d}$	$d \cdot r$	XOR(dr)

Private inference in `tipp++`

- Execution time statistics for different knowledge base access mechanisms, five runs, each accessing the knowledge bases as first time (i.e., case Non-cached).

KB	Mean	St. dev.	Min	Max
baseline	0.17787843	0.00276265	0.17516215	0.18181177
bucket	0.16892560	0.00250258	0.16619691	0.17300868
Anon k4	0.17944365	0.00301458	0.17585521	0.18262578
Anon k6	0.17852928	0.00138278	0.17638855	0.17977013
Anon k8	0.18061575	0.00335783	0.17653908	0.18402709
Anon k10	0.17887916	0.00275545	0.17525422	0.18126626
it2pir	0.22309446	0.00500567	0.21711705	0.22866296

Memory poisoning attacks in MAS

Semantic memory protection

Private local inference with external PIR

What is PIR?

Thanks

Thanks

References

- Torra, V., Bras-Amorós, M. (2026) Memory poisoning and secure multi-agent systems.
  https://arxiv.org/abs/2603.20357
- Torra, V., Bras-Amorós, M. (2026) On Private Knowledge Retrieval and Private Inference for secure multi-agent systems, DPM 2026.