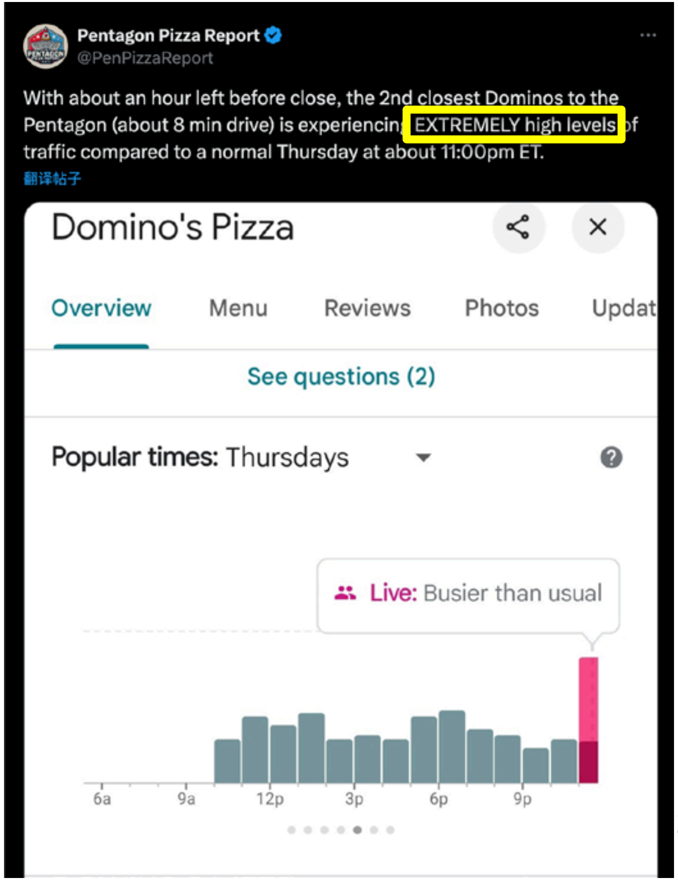
Side Channel Attacks on LLMs

Xiaobei Yan
Nanyang Technological University, CCDS

23-Jul-2025

Did Pizza Predict the Attack on Iran?

- Pizza side-channel attack?



What is Side-Channel Attack

- Unintended information leaks
  - Secret-dependent pattern
  - Medium: Power; Electromagnetic; Cache; Memory; Time; Network or PCIe traffic, etc.

- Passive and active attack
  - Power analysis attack (passive)
  - Fault injection attack (active)

Targets of Side-Channel Attack (SCA)

- SCAs on Cryptosystems
  - Full / Partial key extraction

- SCAs on DNNs
  - Model architecture extraction
  - Model weight extraction
  - Input recovery

- SCAs on LLMs
  - Prompt inversion
  - Response recovery

Content

Title	Side Channel	Date	Venue
What Was Your Prompt A Remote Keylogging Attack on AI Assistants	Network	2024	Usenix
I Know What You Asked Prompt Leakage via KV-Cache Sharing in Multi-Tenant LLM Serving	Time	2025	NDSS
I Know What You Said Unveiling Hardware Cache Side-Channels in Local Large Language Model Inference	Cache	2025	arxiv

Content

Title	Side Channel	Date	Venue
What Was Your Prompt A Remote Keylogging Attack on AI Assistants	Network	2024	Usenix
I Know What You Asked Prompt Leakage via KV-Cache Sharing in Multi-Tenant LLM Serving	Time	2025	NDSS
I Know What You Said Unveiling Hardware Cache Side-Channels in Local Large Language Model Inference	Cache	2025	arxiv

Background – Tokens & LLM Assistants

- Background
  - User sends queries to online LLM assistants
  - Online LLM assistants sends back the response

- Tokenization:
  - Example: "I have an itchy rash." → Tokens: ["I", " have", " an", " itchy", " rash", "."].
  - Spaces/punctuation are often separate tokens.

- LLM Response Generation:
  - Streamed **token-by-token** over encrypted channels (QUIC/TLS).
  - **Side Channel:** Packet size leaks token lengths ($t_i = |r_i|$).

Threat Model

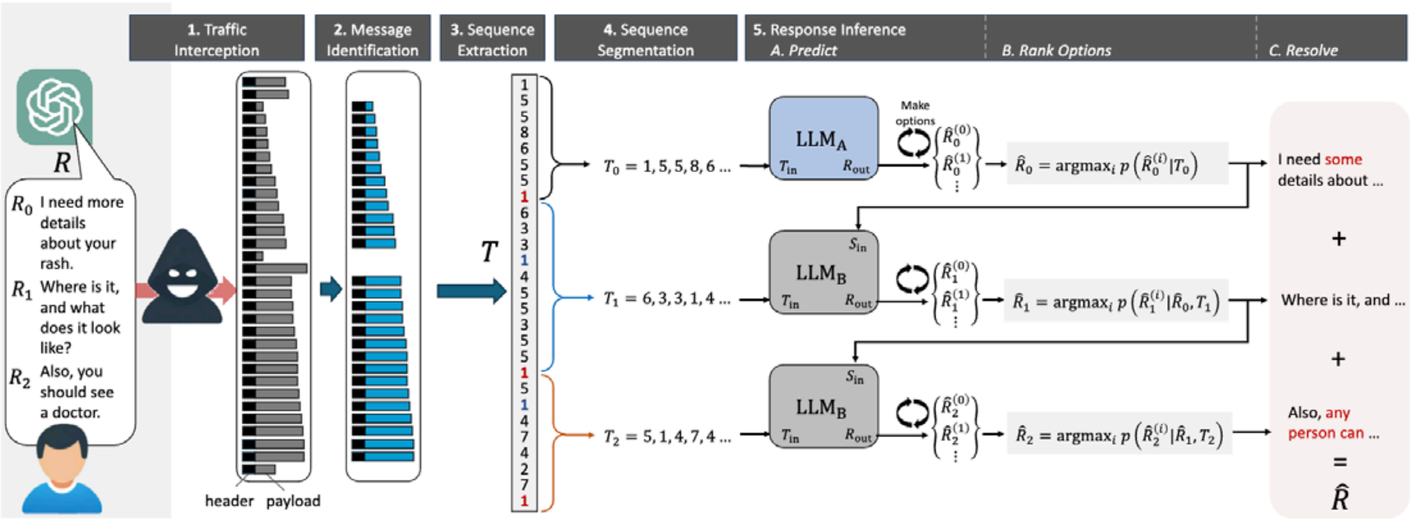
- Attacker Capabilities
  - Network Access:
    - Can extract packet sizes but **not** decrypt content.
  - Knowledge:
    - Knows target service’s protocol (e.g., OpenAI uses QUIC).
    - Has access to public LLM responses (for fine-tuning).
  - Limitations:
    - No access to prompts or model internals (e.g., token probabilities).
    - Assumes no packet padding/compression.

Side-Channel Leakage

- Leakage from side-channel
  - Token length sequence $\mathrm{T} = [\mathrm{t}_1, \mathrm{t}_2, \ldots, \mathrm{t}_\mathrm{n}]$

- Problem Statement
  - Given $\mathrm{T} = [\mathrm{t}_1, \mathrm{t}_2, \ldots, \mathrm{t}_\mathrm{n}]$, infer the response token sequence $\mathrm{R} = [\mathrm{r}_1, \ldots, \mathrm{r}_\mathrm{n}]$, such that $t_i = |r_i|$

Attack workflow



Interception: Attacker captures encrypted packets of *R*.

Message identification: Fixed-size preamble of 4200 bytes and 71 bytes message size.

Token-Length Extraction: Computes $t_i = |m_i| - |m_{i-1}|$ (token lengths).

Segmentation: Uses fine-tuned LLM to map $T = [t_1, t_2, \ldots, t_n] \rightarrow R$.

Prompt Inference: Deduces P from R (e.g., R = *"To treat a rash..."* → P ≈ *"How to treat a rash?"*).

Response Recovery

- Two LLMs for Response Recovery
  - LLM A: generate R_0
  - LLM B: generate R_{i-1}
- Use T5 for recovery
  - encoder-decoder architecture
- Training
  - R_0 = “I need more details about your rash.”

LLM_A Training Prompt

Translate the Special Tokens to English.
Special Tokens: _1 _5 _5 _8 _6 _5 _5 _1

However, a prompt to train LLM_B on R_1 = “*Where is it,* *and what does it look like?*” take the form of:

LLM_B Training Prompt

Translate the Special Tokens to English, given the context.
Context: I need more details about your rash.
Special Tokens: _5 _3 _3 _1 _4 _5 _5 _3 _5 _5 _1

Evaluation

- Datasets & Training:
  - Source: 570K GPT-4 responses from UltraChat
- Split:
  - Training: 550K responses
  - Test: 10K responses
- Models:
  - LLM_A (First Sentence): T5 fine-tuned for 50 epochs.
  - LLM_B (Subsequent Sentences): T5 fine-tuned for 40 epochs.
- Hardware: NVIDIA RTX 6000 (~12 days total training).
- Metrics
  - Cosine Similarity (Φ)
  - ROUGE Scores (R1)
  - Edit Distance (ED)
  - ASR

Evaluation

Real-World Performance

	Vendor	Model	Service	ASR	$\phi > 0.9$	$\phi = 1.0$	R1 >= 0.9	R1 = 1.0	ED <= 0.1	ED = 0.0
No Buff.	OpenAI	GPT-4	in-browser	38.21	15.64	4.57	12.94	5.75	16.20	3.68
	OpenAI	GPT-4	marketplace	53.01	25.80	13.01	28.09	17.02	27.29	10.21
	OpenAI	GPT-4	API	17.69	5.06	0.82	2.65	0.99	2.40	0.57
	Microsoft	Copilot	in-browser	40.87	17.42	7.96	17.96	10.80	17.11	0.51
Buffering	OpenAI	GPT-4	in-browser	35.55	13.70	3.60	10.98	4.79	13.88	2.97
	OpenAI	GPT-4	marketplace	50.28	22.89	10.84	24.03	14.47	23.52	8.56
	OpenAI	GPT-4	API	17.69	5.06	0.82	2.65	0.99	2.40	0.57
	Microsoft	Copilot	in-browser	30.15	5.93	0.16	6.73	0.19	5.18	0.00

Content

Title	Side Channel	Date	Venue
What Was Your Prompt A Remote Keylogging Attack on AI Assistants	Network	2024	Usenix
I Know What You Asked Prompt Leakage via KV-Cache Sharing in Multi-Tenant LLM Serving	Time	2025	NDSS
I Know What You Said Unveiling Hardware Cache Side-Channels in Local Large Language Model Inference	Cache	2025	arxiv

Background

- **Multi-tenant LLM serving** (e.g., vLLM, SGLang) improves efficiency via **KV-cache sharing** for identical token sequences.
- **Problem:** KV-cache sharing introduces **side-channel vulnerabilities**, enabling **prompt leakage** between users.
- **Goal:** Demonstrate how attackers can reconstruct prompts via **KV-cache side channels**.

Background – KV Cache in LLMs

- **KV Cache:** Stores intermediate computations for tokens to speed up inference.
  - Same prefix tokens → same KV cache.
  - Example:
    - User A: *"How to install Windows"*
    - User B: *"How to install Linux"* → Reuses KV cache for *"How to install"*.
- Multi-tenant Scheduler:
  - **Longest Prefix Match (LPM)**
  - First-In-First-Out (FIFO)

Threat Model

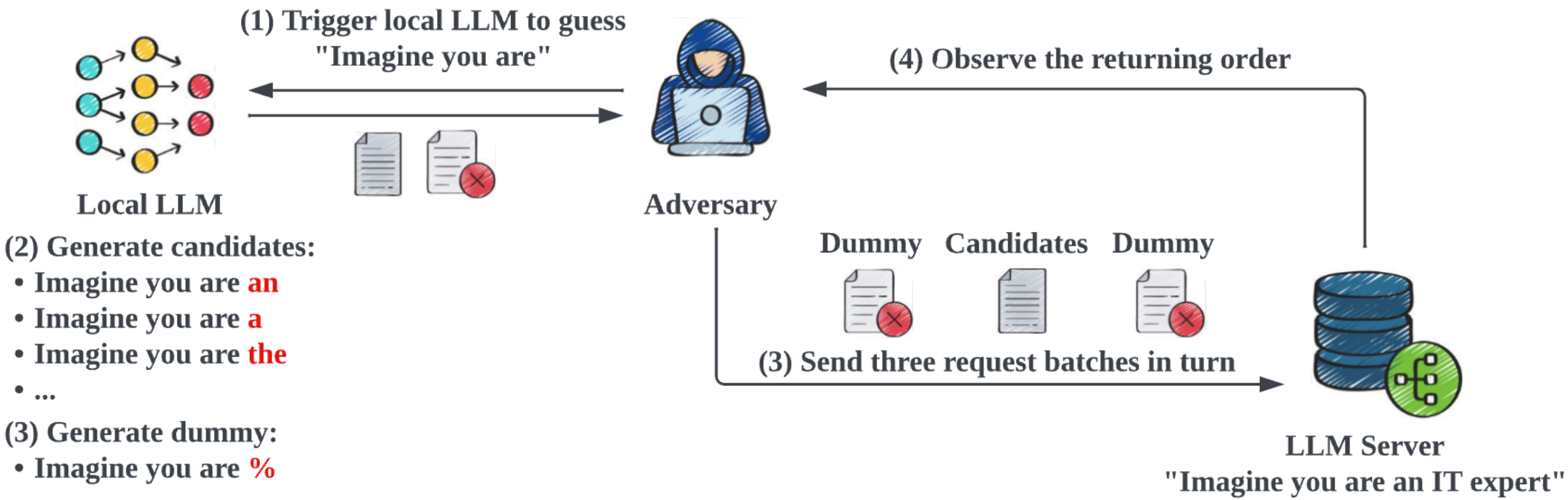
- Adversary Capabilities:
  - Non-privileged user.
  - Knows LLM tokenizer and scheduling and eviction policies (LPM, LRU).

- Attack Goals**:**
  - Reconstruct prompts from other users.

Side-Channel Leakage

- Side-Channel Source: Request Scheduling Order
- Longest Prefix Match (LPM) Policy:
  - Requests with longer matching prefixes get prioritized.
  - Example:
    - Victim’s prompt: *"How to install Windows"*
    - Attacker’s request: *"How to install Linux"* → KV-cache hit for *"How to install"*.
    - Result: Attacker’s request jumps ahead in scheduling queue.

Attack Overview



Token Extraction Mechanism

- Candidate Generation:
  - Local LLM predicts likely next tokens (e.g., *"Imagine you are [an/a/the]"*).
    - $\text{Candidates} = \text{TopK}\big(\text{LLM}(\text{Prefix})\big)$
  - Dummy token (e.g., *"%"*) for baseline comparison.
- Side-Channel Detection:
  - Send [Dummy, Candidates ,Dummy]
  - If **match**: Order = [Dummy, Matched Candidate, Dummy, Unmatched Candidate].
  - If **no match**: Order = [Dummy, Dummy, Candidates].

Stored KV cache:

```
Imagine you are an IT expert  (from victim)
Imagine you are %  (from dummy requests)
```

Waiting queue:

```
Imagine you are %  +
Imagine you are %  |
Imagine you are %  |-Pre
Imagine you are %  |
Imagine you are %  +
Imagine you are a  +
Imagine you are an |-Cands
Imagine you are the+
Imagine you are %  +
Imagine you are %  |
Imagine you are %  |-Post
Imagine you are %  |
Imagine you are %  +
...
```

(a) Serving order before LPM.

Waiting queue:

```
Imagine you are %  +
Imagine you are %  |
Imagine you are %  |-Pre
Imagine you are %  |
Imagine you are %  +
Imagine you are an +-Match
Imagine you are %  +
Imagine you are %  |
Imagine you are %  |-Post
Imagine you are %  |
Imagine you are %  +
Imagine you are a  +
Imagine you are the|-Cands
...
```

(b) Serving order after LPM.

Evaluation

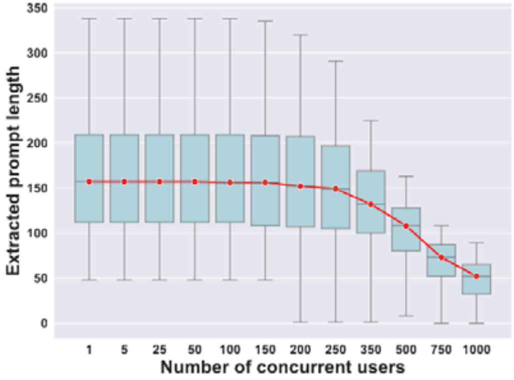


Figure 10: Impact of concurrency level.

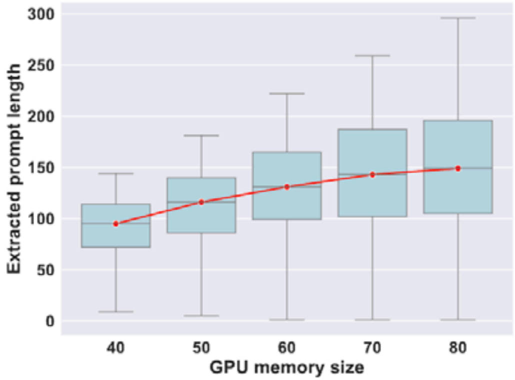


Figure 11: Impact of memory capacity.

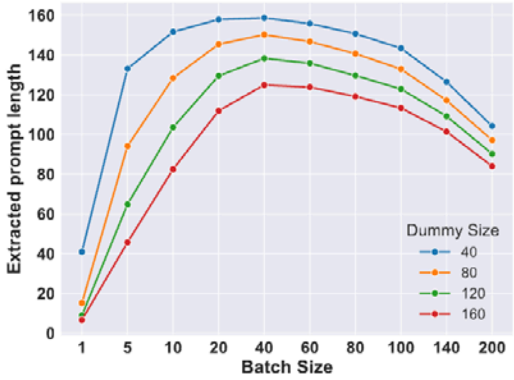


Figure 12: Impact of attack requests.

	Whole Prompt Extraction							Input Extraction							Template Extraction						
	Succ.	Part.	Fail	SR	RR	Req. /inp	Req. /tok	Succ.	Part.	Fail	SR	RR	Req. /inp	Req. /tok	Succ.	Part.	Fail	SR	RR	Req. /inp	Req. /tok
cloze	56	102	22	87%	64%	4843	212	170	4	6	96%	98%	3115	132	102	78	10	94%	77%	4641	59
role	120	33	0	100%	87%	1502	126	151	2	0	100%	99%	1234	68	150	3	0	100%	99%	1687	21
instruction	899	101	0	100%	93%	2183	172	997	3	0	100%	99%	948	50	995	5	0	100%	99%	1298	18

success rate (**SR**)
reversal ratio (**RR**)
(i.e., extracted length / total length)

the average number of requests to extract the entire input (**Req./inp**)
the average number of requests to extract one token (**Req./tok**)

Content

Title	Side Channel	Date	Venue
What Was Your Prompt A Remote Keylogging Attack on AI Assistants	Network	2024	Usenix
I Know What You Asked Prompt Leakage via KV-Cache Sharing in Multi-Tenant LLM Serving	Time	2025	NDSS
I Know What You Said Unveiling Hardware Cache Side-Channels in Local Large Language Model Inference	Cache	2025	arxiv

Introduction

- Hardware cache side-channels can leak sensitive input/output text during LLM inference.
  - The embedding operation creates **secret-dependent** data access
  - The **timing** of embedding operations correlates with the position of the output token

- Goal
  - Reconstruct LLM inputs/outputs via cache access patterns

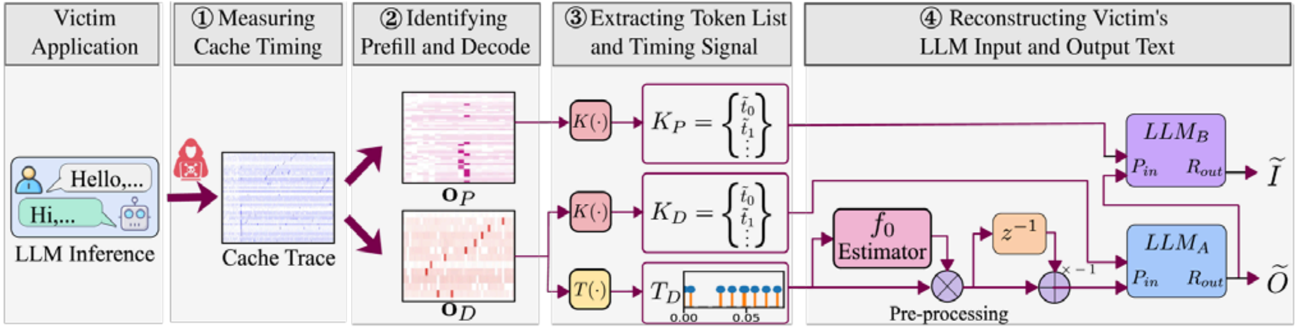
Threat Model

- Adversary Capabilities:
  - Unprivileged spy process **co-located** on victim's machine.
  - **Passive cache monitoring**: No direct interaction with victim LLM;.
  - **Flush+Reload**: Accesses shared memory (via page cache or page deduplication).
- Victim Scenario:
  - User interacts with **local** LLM (e.g., confidential emails, personal advice).
  - Token embedding operations **offloaded** to CPU.

Side Channel Leakage

- Token Value Leakage:
  - Cache access patterns during token embedding reveal token indices.
    - $\boldsymbol{E} = \boldsymbol{W}\boldsymbol{x}, where\ \boldsymbol{x_i} = \left[0, \ldots, 1_{t_i}, \ldots, 0\right]^T$
  - Autoregressive decoding leaks both input and output tokens.
- Token Position Leakage:
  - Timing of decode phases exposes token order.
- Challenges:
  - Noise in cache traces (false positives/negatives).
  - Shuffled input tokens (prefill phase) due to parallel processing.

Attack Workflow



1.Cache Trace Collection: Spy process monitors shared cache.

2.Phase Identification: Separates prefill (batched) and decode (serial) phases.

3.Token Mapping: Converts cache hits to token lists and timing signals.

4.Output Reconstruction: Uses fine-tuned LLM (LLM_A) to denoise and reconstruct response.

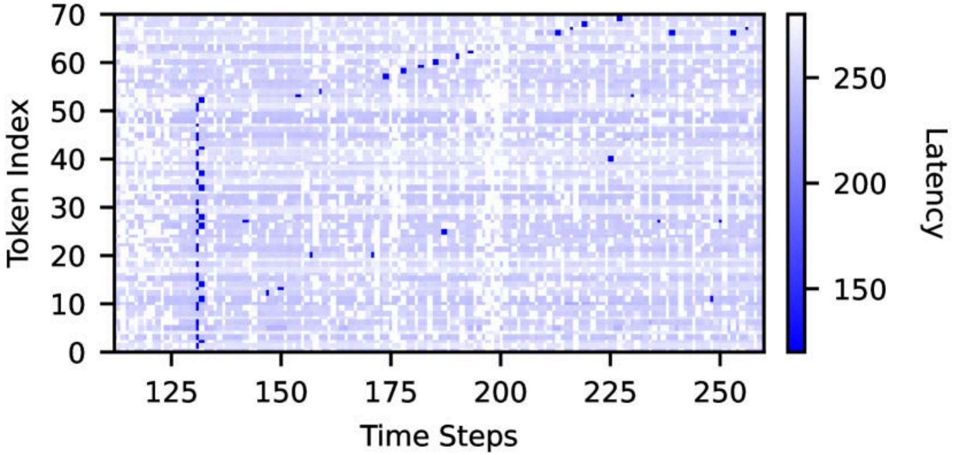
5.Input Reconstruction: Leverages context with output to restore shuffled input tokens (LLM_B).

Example of Cache Trace

The obtained cache trace $\mathbf{o}$ is a $L \times |V|$ matrix.

L: cache trace length.

$|V|$: vocabulary size.



Measuring Cache Time

- Start address calculation
  - $p_1 + p_2 + iDb \le A_i < p_1 + p_2 + (i+1)Db$
  - p_1**:** start address p_2**:** embedding table $\boldsymbol{W}$ offset
  - D: dimension of the embedding table; b size of the vector element.
- Evading Hardware Prefetchers
  - Array-of-Pointers (AoP) prefetchers introduce high noise
  - Add an offset to pointers to prevent prefetch

Evaluation

- LLM A and LLM B:
  - GPT-4o-mini-2024-07-18

- Metrics
  - Average number of input and output tokens (N_o)
  - ROUGE Scores ($R1, RL$) Levenshtein Similarity (LS)
  - Cosine similarity (Φ)
  - ASR: the proportion of testing samples where $\Phi > 0.77$.

Victim LLM	Dataset	Output Reconstruction						Input Reconstruction					
		N_O	R1 (%)	RL (%)	LS (%)	ϕ (%)	ASR (%)	N_I	R1 (%)	RL (%)	LS (%)	ϕ (%)	ASR (%)
Google Gemma2-9B [4]	UltraChat	243	98.2	98.2	97.0	99.6	99.8	20	93.5	90.2	87.4	99.2	**100.0**
	NQ-Open	79	95.9	95.9	94.3	98.7	99.3	13	94.6	93.0	91.3	99.0	**100.0**
	SIQA	193	96.4	96.4	94.2	98.8	99.1	31	86.6	79.2	74.8	96.9	**100.0**
	SQuAD2	55	91.5	91.5	89.8	98.2	**100.0**	183	57.1	47.7	34.4	94.9	**100.0**
	ChatGPT-Roles	222	98.7	98.7	98.0	99.6	**100.0**	48	85.4	79.7	70.6	99.1	**100.0**
Meta Llama-3.1-8B [8]	UltraChat	253	99.0	99.0	98.9	99.2	99.3	19	94.5	91.9	89.5	99.2	**100.0**
	NQ-Open	162	97.4	97.4	96.9	98.1	98.0	12	**94.8**	**93.4**	**91.4**	99.0	**100.0**
	SIQA	64	98.1	98.1	97.6	98.9	99.1	30	86.1	78.5	73.6	96.6	99.7
	SQuAD2	20	90.1	90.1	90.4	96.7	96.4	180	55.8	46.4	33.2	94.3	**100.0**
	ChatGPT-Roles	215	**99.5**	**99.5**	**99.6**	**99.8**	**100.0**	48	86.3	80.7	72.3	99.0	**100.0**
TII Falcon3-10B [3]	UltraChat	175	98.4	98.4	97.3	99.6	99.6	20	**94.8**	92.1	90.2	**99.3**	**100.0**
	NQ-Open	109	98.2	98.1	97.7	99.7	99.9	13	94.3	92.6	91.1	99.0	**100.0**
	SIQA	140	98.9	98.9	97.9	99.7	**100.0**	31	86.2	78.6	75.5	96.7	**100.0**
	SQuAD2	62	90.6	90.6	93.2	98.0	96.4	185	54.6	44.9	33.5	93.8	**100.0**
	ChatGPT-Roles	67	98.9	98.8	99.3	99.6	**100.0**	48	86.8	82.3	73.9	99.0	**100.0**
Mistral-7B [13]	UltraChat	256	94.6	94.6	91.6	98.2	98.7	20	91.6	87.7	84.4	98.7	**100.0**
	NQ-Open	120	95.1	95.1	94.6	97.1	96.8	12	89.1	84.0	80.8	97.3	99.8
	SIQA	65	98.7	98.7	98.2	99.4	99.7	32	85.9	77.6	73.7	96.2	**100.0**
	SQuAD2	57	91.4	91.4	90.1	96.9	98.2	204	51.3	43.2	32.4	92.7	98.2
	ChatGPT-Roles	243	94.6	94.6	91.6	98.9	**100.0**	54	83.2	78.4	69.7	97.9	**100.0**
Microsoft Phi-3.5-mini-3B [12]	UltraChat	263	93.5	93.5	88.9	99.0	**100.0**	21	90.5	87.2	84.3	98.2	99.6
	NQ-Open	194	93.9	93.9	90.9	98.7	99.3	12	88.0	82.9	79.8	97.0	99.8
	SIQA	253	92.7	92.7	87.7	98.5	99.4	33	85.2	78.5	75.4	96.5	99.7
	SQuAD2	137	93.5	93.5	90.6	97.6	98.2	209	51.0	42.4	32.2	92.1	96.4
	ChatGPT-Roles	263	94.6	94.6	92.1	98.8	**100.0**	57	80.6	75.0	65.7	97.6	**100.0**
Average		165	96.3	96.3	94.8	98.7	99.1	24	89.9	85.8	82.7	98.0	99.9

Future Research

- Attack surface
  - LLM / Agent / RAG

- Attack target
  - **User / System prompt**
  - **Response**
  - What other information is worth stealing?
    - PII
    - API key
    - Chain of thought / actions
    - ……

- Attack practicability
  - Multi-tenant LLM architecture / co-locate with victim $\rightarrow$ remote?

Thank You!