Stealing Watermarks of Large Language Models via Mixed Integer Programming

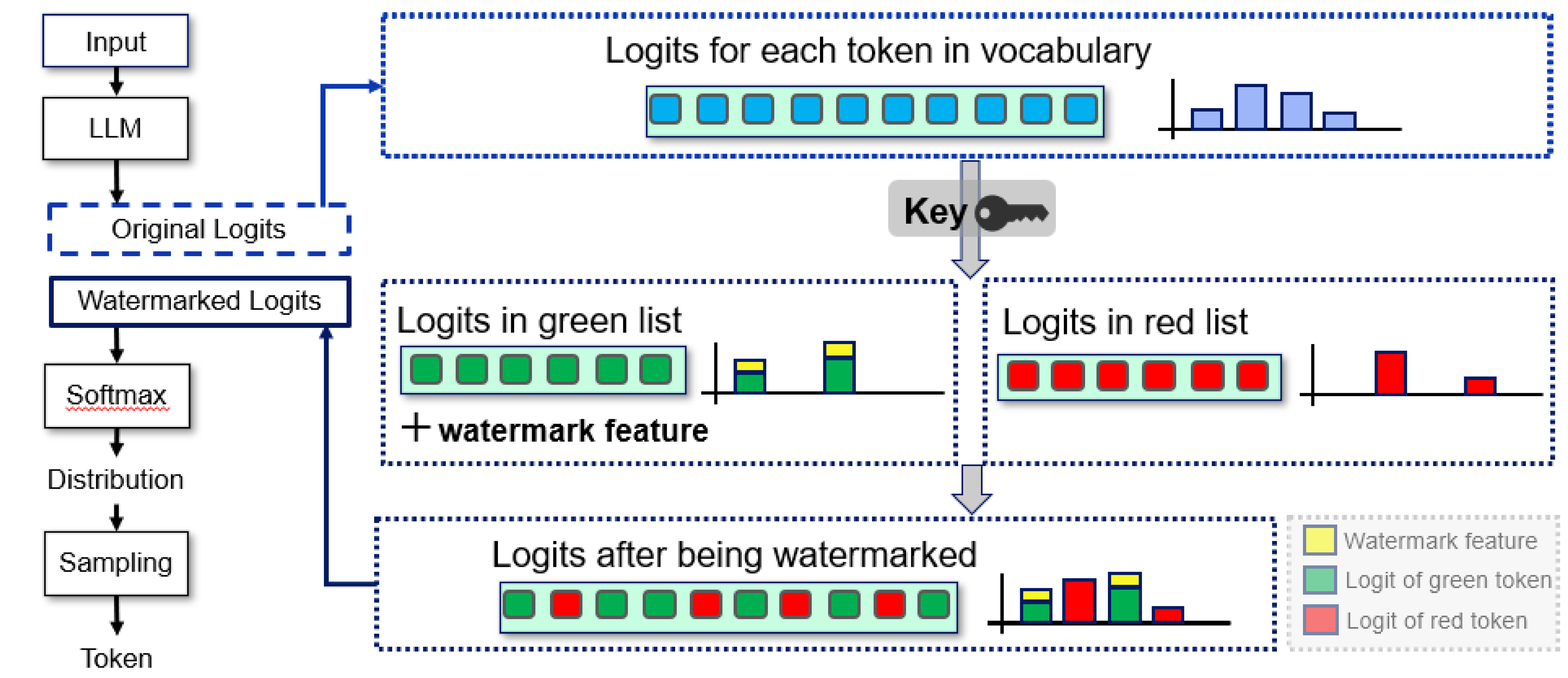
Zhaoxi Zhang, Xiaomei Zhang, Yanjun Zhang, Leo Yu Zhang, Chao Chen, Shengshan Hu, Asif Gill, Shirui Pan
University of Technology Sydney, Griffith University, Royal Melbourne Institute of Technology, Huazhong University of Science and Technology

1. INTRODUCTION

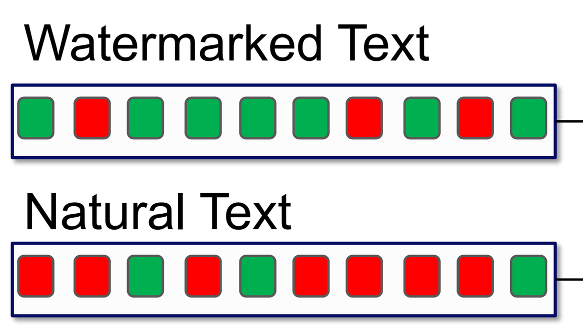
- LLM watermark is a novel technique to address copyright concerns, monitor AI-generated text, and prevent misuse.
- A watermark stealing attack aims to infer the details of an LLM watermarking scheme.
- Attackers can model watermark stealing as a constrained optimization problem to extract and remove the watermark without affecting text semantics.
- The proposed method can also be extended to handle multi-key watermark scenarios.

2. BACKGROUND: LLM WATERMARK

- **Adading Watermark into LLM**:
  Randomly divide the vocabulary into a green list and a red list, then raise the sampled probability by adding watermark features into the green list.



- **Watermark Detection**:
  After watermarking, the number of green tokens in the watermarked text is greater than in the non-watermarked text:



- Count the number of green tokens $G(S)$
- $G(S_{wm}) > G(S_{natural})$

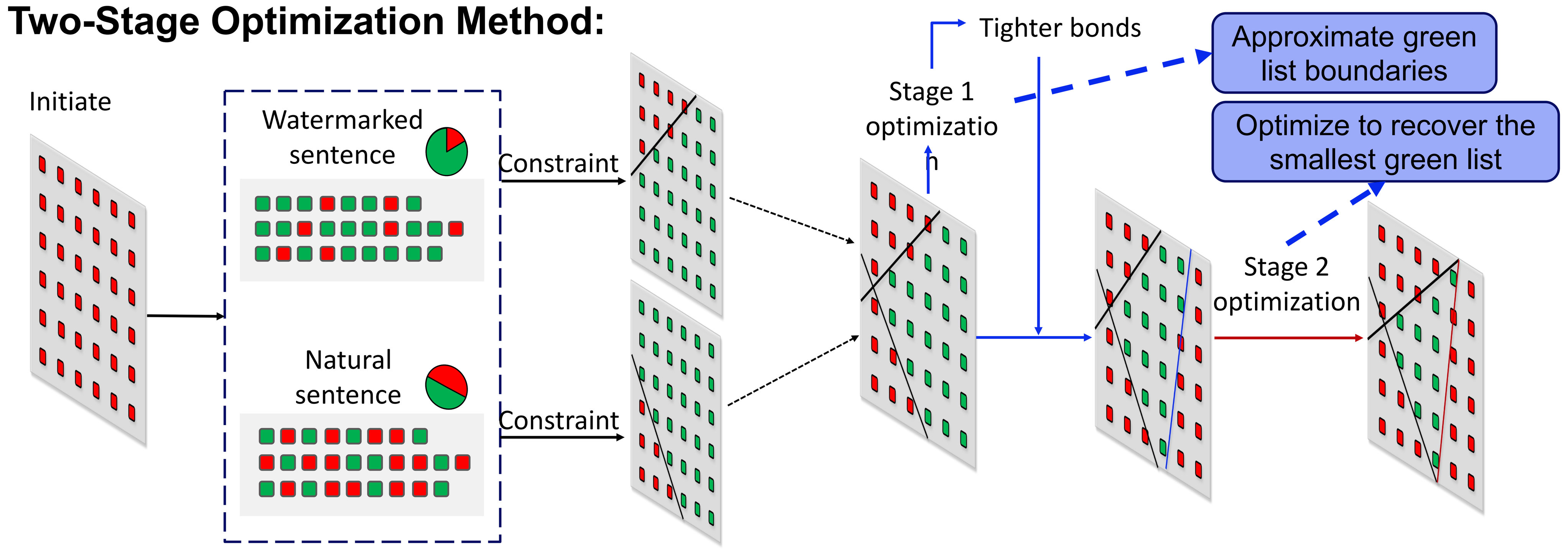
3. OUR METHODS: INSIGHTS

Watermark Stealing as an Optimization Problem:

- The association between tokens and the green list can be represented as integers;
- Constraints: watermark detection rules;
- Objective: finding a minimal available green list for the watermark text.

4. WATERMARK STEALING

- **Threat Models:**
  Attack Setting 1 (AS1) → attackers ***can*** access the watermark detector API.
  Attack Setting 2 (AS2) → attackers ***cannot*** access the watermark detector API.
- **Two-Stage Optimization Method:**

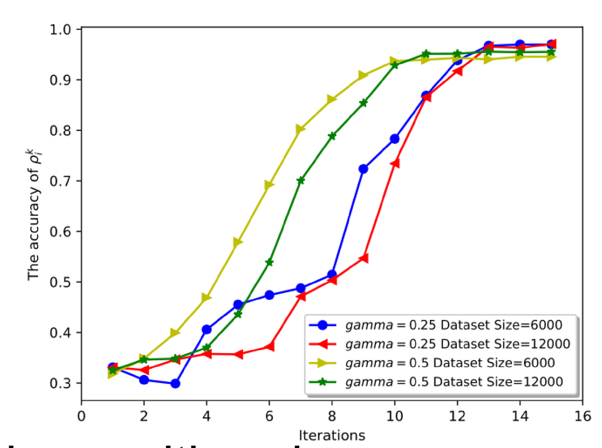


- **Multi-Key Stealing**: In this scenario, the attacker need to find suitable key for each sentence.

Random initialize $\{\hat{\rho}_i^k\}$
for until optimization converge **do**
  Optimize Stage-1 while fixing $\hat{\rho}_i^k$;
  Optimize Stage-2 while fixing $\hat{\rho}_i^k$;
  Resign $\hat{\rho}_i^k = \begin{cases} 1, \text{if } k = \operatorname{argmax}_k(G(S_i, k)) \\ 0, \text{else} \end{cases}$
end for

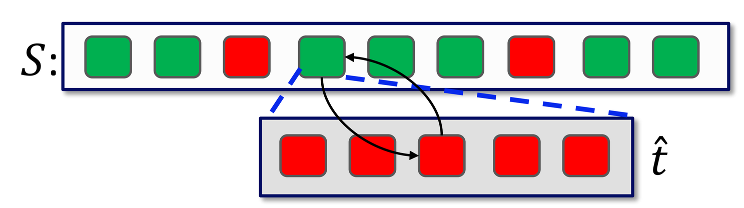
Which key is suitable for this sentence

The number of green token under key k



The accuracy of ρ while iterative optimization

- **Removing watermarks** in sentences by replacing green tokens with red ones.



Green token · Sentence S
Red token · Candidate Red token $\hat{t}$

5. RESULTS

- **Experimental Settings**: LLM: OPT-1.3B, LLaMA-2-7B; Dataset: C4; Solver for the Mixed Integer Programming: Gurobi.
- **Stealing performance** against LLaMA-2-7B under AS1 and AS2

Watermark setting	Dataset size	Ours (AS1) N_g	Ours (AS1) N_t	Ours (AS1) Precision(↑)	Freq. (AS1) N_g	Freq. (AS1) N_t	Freq. (AS1) Precision(↑)	Ours (AS2) N_g	Ours (AS2) N_t	Ours (AS2) Precision(↑)	Freq. (AS2) N_g	Freq. (AS2) N_t	Freq. (AS2) Precision(↑)
$\gamma = 0.25$, $\delta = 2$	4000	1064	885	83.18%	5154	2547	49.42%	3165	2003	63.29%	6032	2782	46.12%
	10000	1431	1224	85.53%	5519	2970	53.81%	2852	2069	72.55%	6613	3223	48.74%
	20000	1396	1256	89.97%	5494	3181	57.90%	2582	2056	79.63%	6727	3505	52.10%
	40000	2146	1912	89.10%	5425	3335	61.47%	2393	1990	83.16%	6680	3693	55.28%
$\gamma = 0.25$, $\delta = 4$	4000	732	678	92.62%	4350	2867	65.91%	3884	2813	72.43%	4392	2882	65.62%
	10000	780	731	93.72%	4704	3259	69.28%	4466	3347	74.94%	4736	3275	69.15%
	20000	867	803	92.62%	4895	3498	71.46%	4443	3481	78.35%	4937	3517	71.24%
	40000	933	861	92.28%	5020	3737	74.44%	4969	3923	78.95%	5062	3754	74.16%

- N_g: the number of tokens in the stolen green list
- N_t: the number of **true** green tokens in the stolen green list
- Precision= N_g/N_t
- Precision Average higher 18.23% / 9.52% in AS1 / 2

- **Removal performance** against LLaMA-2-7B under AS1 and AS2

Watermark setting	Dataset size	AS1 G_{avg}^b	AS1 $G_{avg}^a(\downarrow)$ Ours	AS1 $G_{avg}^a(\downarrow)$ Freq.	AS1 GRR(↓) Ours	AS1 GRR(↓) Freq.	AS2 G_{avg}^b	AS2 $G_{avg}^a(\downarrow)$ Ours	AS2 $G_{avg}^a(\downarrow)$ Freq.	AS2 GRR(↓) Ours	AS2 GRR(↓) Freq.
$\gamma = 0.25$, $\delta = 2$	4000	68.01	11.24	21.54	28.55%	52.56%	71.17	10.38	36.62	14.58%	51.46%
	10000	68.01	11.17	19.89	21.19%	50.84%	71.17	9.62	35.84	13.52%	50.35%
	20000	68.01	8.19	19.27	21.05%	50.37%	71.17	9.53	35.10	13.40%	49.32%
	40000	68.01	8.42	18.80	13.44%	50.41%	71.17	9.64	34.90	13.55%	49.04%
$\gamma = 0.25$, $\delta = 4$	4000	52.45	7.12	15.02	31.11%	47.81%	71.13	8.32	34.36	11.70%	48.30%
	10000	52.45	6.63	13.66	29.42%	47.49%	71.13	7.45	34.09	10.47%	47.92%
	20000	52.45	6.47	13.17	29.34%	48.35%	71.13	7.38	34.63	10.38%	48.68%
	40000	52.45	6.45	12.91	28.97%	48.81%	71.13	7.58	34.88	10.66%	49.04%

- G_{avg}^b: average number of green tokens ***before*** removal
- G_{avg}^a: average number of green tokens ***after*** removal
- GRR= G_{avg}^a/G_{avg}^b: the rate of remaining green tokens
- GRR Average lower 29.98% / 38.81% in AS1 / 2

https://github.com/plll4zzx/mip_watermark_stealing

UTS UTS CRICOS PROVIDER CODE: 00099F / UTS TEQSA PRV12060