PRIVMARK: Private Large Language Models Watermarking with MPC

Thomas Fargues[†,*], Ye Dong[‡,✉], Tianwei Zhang[§], Jin-Song Dong[‡]
[†]Télécom SudParis, France, [‡]National University of Singapore, [§]Nanyang Technological University, Singapore
thomas.fargues@telecom-sudparis.eu, dongye@nus.edu.sg, tianwei.zhang@ntu.edu.sg, dcsdjs@nus.edu.sg

***Abstract*—The rapid growth of Large Language Models (LLMs) has highlighted the pressing need for reliable mechanisms to verify content ownership and ensure traceability. Watermarking offers a promising path forward, but it remains limited by privacy concerns in sensitive scenarios, as traditional approaches often require direct access to a model's parameters or its training data. In this work, we propose a secure multi-party computation (MPC)-based private LLMs watermarking framework, PRIVMARK, to address the concerns. Concretely, we investigate PostMark (EMNLP'2024), one of the state-of-the-art LLMs Watermarking methods, and formulate its basic operations. Then, we construct efficient protocols for these operations using the MPC primitives in a black-box manner. In this way, PRIVMARK enables multiple parties to collaboratively watermark an LLM's output without exposing the model's weights to any single computing party. We implement PRIVMARK using SecretFlow-SPU (USENIX ATC'2023) and evaluate its performance using the ABY3 (CCS'2018) backend. The experimental results show that PRIVMARK achieves semantically identical results compared to the plaintext baseline without MPC and is resistant against paraphrasing and removing attacks with reasonable efficiency.**

***Index Terms*—Privacy, Security, Large Language Models, Watermarking, Secure Multi-Party Computation**

I. INTRODUCTION

Large Language Models (LLMs) such as ChatGPTT [1] have demonstrated remarkable capabilities in applications ranging from content creation to conversational agents. As these models become more powerful and integrated into commercial and critical systems, questions of authorship, intellectual property, and misuse become increasingly salient [2]. Malicious actors can exploit these models to generate misinformation, plagiarized content, or other harmful text, making it difficult to trace the origin of the generated output. Consequently, developing reliable techniques to identify machine-generated text and verify model ownership is a critical area of research. Digital watermarking [3], [4] offers a compelling solution to these challenges. By subtly embedding a secret signal, a watermark, into the generated text, model owners can later prove their ownership or identify the source of a particular piece of content. Watermarking techniques fall into two categories: modifying training/architecture (e.g., fine-tuning to embed a detectable dialect [5]) or applying watermarks during/after generation. Commonly, inference-time methods partition vocabulary into green and red lists, steering generation toward green tokens to induce detectable bias [6], [7]. Post-generation watermarking schemes, such as the PostMark [8], are particularly attractive as they do not require costly retraining and can be applied to pre-trained LLMs directly.

However, privacy remains a barrier, as the LLM, data, and keys are often sensitive. For example, in collaborative settings where companies co-own models without sharing data, traditional methods requiring centralized access are not viable.

Our Contributions. To bridge this gap, we propose the first, to our best knowledge, private LLM watermarking framework, PRIVMARK, by integrating Secure Multiparty Computation (MPC) [9], [10] primitives into LLMs watermarking progress to protect privacy. MPC is a cryptographic technology that enables multiple parties to jointly compute a function over their private inputs without revealing those inputs to one another. Concretely, we secret-share the private LLM weights, watermarking parameters, and user data among different non-colluding computing parties. By making use of advanced three-party (3PC) primitives in the Arithmetic Black Box (ABB) model [11], we construct efficient 3PC protocols for watermark insertion and detection. Our PRIVMARK represents a significant step towards enabling secure and accountable LLMs. In summary, our contributions are threefold:

- We conduct a systematic investigation and comparison of existing LLM watermarking approaches. For generality, we employ PostMark [8], one state-of-the-art technique, as the representative watermarking method.
- We decompose PostMark into fundamental operations to develop efficient, secure protocols leveraging 3PC primitives within the ABB model.
- We implement our protocol in SecretFlow-SPU via ABY3 [12]. Evaluations confirm feasibility, privacy preservation, and utility. PRIVMARK achieves comparable robustness against paraphrasing and removal attacks §V-C with reasonable overhead.

Organization. In §II, we summarize related works. Then, we introduce PostMark and 3PC in §III. The concrete design of PRIVMARK is shown in §IV. Finally, we show experimental evaluations in §V and conclude this work in §VI.

II. RELATED WORK

We summarize recent watermarking approaches and MPC-based private inference frameworks for LLMs.

*Work during internship at Nanyang Technological University, Singapore.
✉Corresponding author.

A. Watermarking of Large Language Models

The proliferation of powerful LLMs has spurred research into techniques for identifying machine-generated text, which can be applied during training or post-generation [13]. Training-time methods embed a signal by modifying the model's architecture or fine-tuning process, but this is often computationally expensive and impractical for proprietary, pre-trained models. Post-generation (or black-box) watermarking methods are more flexible, as they operate on the model's output probabilities without requiring access to its internal weights. Also, this method is more efficient [13]. As shown in Table I, we give a holistic comparison of representative methods in terms of fidelity, robustness, efficiency, and undetectability, which are the most important properties of Watermarking methods.

TABLE I: Comparison of LLM watermarking Methods.

Method	Fidelity	Robustness	Efficiency	Undetectability
Unicode [14]	★★★ Preserves semantics	★ Very Weak	★★★ Extremely High	★★ Machine-detectable
Yang et al. [15]	★★★ High Fidelity	★★ Moderate-to-High	★★ Moderate	★★★ Imperceptible
DeepTextMark [16]	★★★ High Fidelity	★★★ High	★★ Good	★★★ Imperceptible
PostMark [8]	★★ Medium	★★★★ Very High	★ Lower	★★★ Imperceptible
REMARK-LLM [17]	★★★ High Fidelity	★★★ High	★★ Moderate	★★★ Imperceptible

From our analysis, three methods emerged as strong candidates with high resilience to paraphrasing attacks: DeepTextMark [16], PostMark [8], and REMARK-LLM [17]. However, the PostMark algorithm [8] is a prominent example of a robust black-box technique. It works by using a secret table to pseudo-randomly select words to insert into the text of the embedded watermark. This makes PostMark an ideal candidate for pioneering the integration of watermarking into a privacy-preserving scenario.

B. MPC-based Private LLMs Inference

Secure Multi-Party Computation (MPC) provides a cryptographic foundation for parties to jointly compute a function on their private data [9], [10]. Recently, the focus has shifted towards the unique challenges posed by large-scale Transformer architectures. Several works have proposed MPC-based solutions for secure Transformer inference [18]–[23] in two or three-party. These advancements have concentrated almost exclusively on the core task of secure inference.

III. Background

In this section, we introduce PostMark WaterMarking and replicated secret sharing-based three-party computation (3PC).

A. Revisiting PostMark

PostMark [8] is a post-generation watermarking algorithm designed for black-box LLMs. The algorithm consists of two main procedures: insertion and detection.

Insertion. As shown in Figure 1 (a), PostMark embeds watermarks with EMBEDDER by leveraging semantic similarity between the input text and a set of carefully selected watermark words. The process begins by generating a semantic embedding of the original text using a pretrained embedding model (such as OpenAI's text-embedding-3-large). This embedding

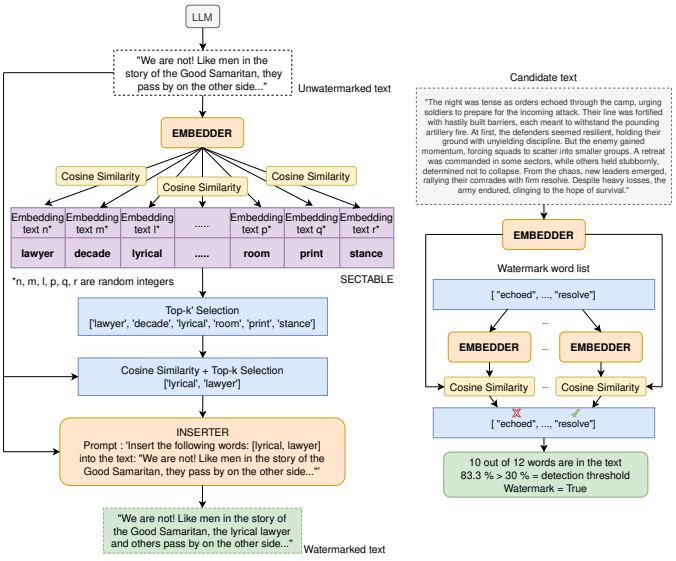


Fig. 1: PostMark watermarking insertion (a) and detection (b).

is compared to a secret word embedding table, SECTABLE[1], which contains embeddings randomly assigned to a curated vocabulary of content words. And after, the top-k similar words from SECTABLE are selected to serve as watermark candidates.

An instruction-following language model (INSERTER), such as GPT-4o, is then prompted to rewrite the original text while incorporating the selected watermark words in a natural and fluent way, preserving the original semantics.

Detection. For detection (Figure 1 (b)), the system re-embeds the suspect text, reconstructs the list of candidate watermark words based on similarity to a candidate list, and checks how many of those words appear in the text. If the overlap exceeds a certain threshold, the text is considered watermarked.

PostMark is designed to be robust against paraphrasing attacks, since it conditions the watermark on the text's semantic embedding, which remains relatively stable under paraphrasing.

B. Secure Multiparty Computation & Threat Model

1) MPC Preliminaries: PrivMark is built upon **replicated secret sharing-based three-party computation (3PC)** [24]. A secret value x is split into three shares, (x_1, x_2, x_3), such that $x = x_1 + x_2 + x_3$ in ring $\mathbb{Z}_{2^\ell}$. These shares are then distributed among three computing parties as: P_1 holds (x_1, x_2), P_2 holds (x_2, x_3), and P_3 holds (x_3, x_1).

This structure allows parties to perform computations on the shares directly. **Linear operations**, such as addition, can be performed locally by each party on their respective shares without any communication. **Non-linear operations**, such as multiplication, require dedicated and efficient interactive protocols where parties exchange messages to compute shares of the product. Existing works [12], [20], [25] have proposed efficient protocols for widely used arithmetic and boolean functions (including comparison, selection, etc) and even secure

[1]This table is built from external files: a vocabulary V and a set of embeddings D, we detail this in §IV-B

inference of neural networks. We use them in the ABB model to construct PRIVMARK.

2) Threat Model: We follow the standard **honest-but-curious** security and build PRIVMARK upon ABY3 protocol. Under this model, all computing parties are expected to honestly follow the protocol's instructions, but they may attempt to learn additional information about other parties' private data from the messages they receive during the computation. Our security guarantees hold as long as at most one of the three parties is corrupted by the passive adversary.

IV. DESIGN OF PRIVMARK

A. Overview of Our Design

To achieve secure watermarking, PRIVMARK uses three non-colluding computing parties to execute ABY3 [24] to insert and detect the watermarking. PRIVMARK is designed to follow the same workflow as PostMark, but with a focus on privacy preservation. The workflow is illustrated in Figure 2:

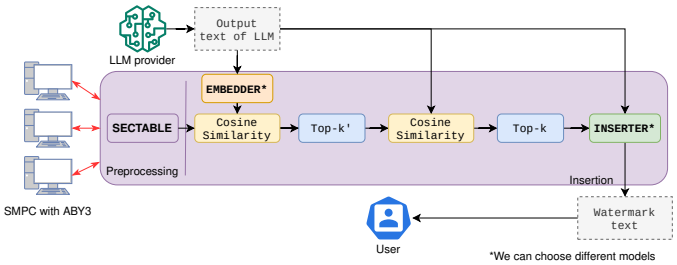
- For secure evaluations of `EMBEDDER` and `INSERTER`, we utilize ABY3-based secure inference solution [20] to achieve secure embedding and LLM-based rewriting.
- For other operations of PostMark, we invoke the 3PC primitives provided by SecretFlow-SPU [25].

The solution can be understood as a workflow involving the client, the LLM provider, and three computing parties that jointly process the watermarking tasks.

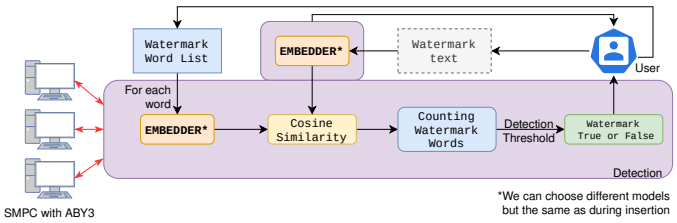
Scenario. Consider a company that develops innovative product designs and relies on an external LLM provider to assist in drafting technical reports and marketing documents. These documents are part of the company's intellectual property (IP) and must be protected against unauthorized redistribution or misuse. With PRIVMARK, the company can ensure that all LLM outputs contain invisible watermarks. Even if competitors or external parties obtain the generated documents, the watermark allows the company to prove ownership and trace the origin of the content. Importantly, with MPC, the watermarking process does not expose the company's sensitive data nor the parameters of the LLMs and watermarking algorithms.

Insertion Workflow. The watermark insertion proceeds as follows: **1) Client request**: The client sends a query to the LLM provider. **2) LLM output**: The LLM provider generates an answer t, which serves as input to PRIVMARK. **3) Initialization**: Three computing parties start the secure computation and jointly generate the `SECTABLE`. **4) Secure embedding**: Using the `EMBEDDER`, the three parties securely embed t as $\langle e_t \rangle$. **5) First similarity search**: The parties compute cosine similarity between $\langle e_t \rangle$ and `SECTABLE`, then select the top-k' candidate words. **6) Watermark selection**: Watermark words are securely selected. Using `INSERTER`, they are injected into t via prompt modification. **7) Delivery**: The final watermarked text is returned to the user. Figure 2 (a) and Algorithm 1 show details.

Detection Workflow. The watermark verification process follows these steps: **1) Client submission**: The user provides PRIVMARK with a candidate text and the list of watermark words. **2) Secure embedding**: The three computing parties embed both the candidate text and watermark word list using the `EMBEDDER`. **3) Similarity computation**: For each word, the three parties compute the cosine similarity securely. **4) Watermark identification**: The system counts how many words match the watermark pattern. **5) Threshold decision**: Based on a detection threshold, PRIVMARK determines whether the candidate text is watermarked. **6) Result delivery**: The decision (watermarked / not watermarked) is returned to the user. All details are shown in Figure 2 (b) and Algorithm 3.



(a)



(b)

Fig. 2: Illustration of PrivMark insertion (a) and detection (b).

B. Watermark Insertion

Embedder. The `EMBEDDER` accounts for converting both individual words and entire documents into a high-dimensional numerical representation. In PRIVMARK, we employ the open-sourced *intfloat/e5-base* [26] for the compilation with SPU.

SecTable: PostMark's central concept is to have an LLM subtly embed a pre-selected list of watermark words into text. This is done without significantly altering the text's quality or meaning. The watermark words themselves are chosen by comparing the text's embedding with a `SECTABLE`, which is a word embedding table, using cosine similarity. `SECTABLE`'s creation involves two key steps, explained further below.

- Step 1: Vocabulary Selection V
  The initial step is to establish a suitable vocabulary, V. This process begins with the WikiText-103 corpus [27] as a foundational vocabulary source. To ensure that the watermarking process does not introduce nonsensical or out-of-place words, a filtering procedure is applied. This refinement involves the removal of: function words, proper nouns, infrequent and rare words. For our experiments §V, we used V provided by PostMark [8].
- Step 2: Mapping Vocabulary to Embeddings
  The second step focuses on mapping the words in V to their corresponding embeddings in a secure manner. To prevent adversarial recovery of the embedding table, `SECTABLE`

is constructed by randomly assigning each word in the vocabulary to an embedding during a prepossessing step. This resulting map effectively serves as a cryptographic key. The procedure is as follows:

1) A set of embeddings, D, is generated by applying an EMBEDDER to a collection of random documents. For our experiments §V, we used a D created with `wikitext-103-raw-v1` [27] and *intfloat/e5-base* [26], but in practice, these can be random 250-word excerpts from a large dataset like the RedPajama dataset as shown in PostMark [8].
2) Each word in the vocabulary V is then randomly mapped to a unique document embedding from the set D. This mapping constitutes the final SECTABLE.

Remark 1: For security, we partition data across the three computing parties: one party stores the embedding set D and another stores the vocabulary V. The parties then jointly and securely compute SECTABLE using ABY3 so that neither D nor V is revealed in the clear. Secure construction of SECTABLE is essential because the table constitutes the scheme's trust anchor and effectively serves as the cryptographic key for the watermarking mechanism; any leakage during its generation would compromise the protection.

It is also important to mention that the party who holds the vocabulary V must be trustworthy because, it stores the watermarking words to insert. A malicious party may generate a vocabulary V that is unusable for the watermarking solution.

Remark 2: Also, a more straightforward approach might be to use the EMBEDDER's inherent word embeddings directly. However, this method is eschewed because it is vulnerable to attacks and has been shown to be less effective. The reduced effectiveness is partly due to the high probability of many potential watermark words already existing in the input text prior to the watermarking process.

Words Selection. Then, we compare ⟨SECTABLE⟩ and the embedding text $\langle e_t \rangle$ with the function *cosine similarity* 1 (l.9). It is a metric used to quantify the semantic similarity between the texts embeddings in this example. The resulting score ranges from -1 (semantically opposite) to 1 (semantically identical), with scores closer to 1 indicating a higher degree of similarity. The formula of *cosine similarity* is :

$$\text{CosSim(A,B)} = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (1)$$

Afterwards, we need to select the *topk* with k' most similar words, where we choose the k' results with the highest score based on the SECTABLE, we pick the top-k' words. Next, we embed the top k' words and compute their cosine similarity with the original text embedding $\langle e_t \rangle$. Finally, we select the Top-k with k results to get our watermark words to insert.

Inserter: The INSERTER's purpose is to link words from the watermark list into the original text by rewriting it, through another LLM. This can be completed by an existing secure inference solution like PUMA [20]. Putting it all together, we give protocol Π_{MarkIn} for watermark insertion in algorithm 1.

Algorithm 1 Protocol Π_{MarkIn} for Watermark Insertion

Require: Party P_1 holds plaintext t, $wordlist$ and LLM parameters. Party P_2 holds $embeddingList$ and embedded parameters. Public insertion ratio $r = 12\%$ as [8].
Ensure: P_1 receives watermarked text t'

$n \leftarrow \text{CountWords}(t)$
$k \leftarrow \text{NbWatermarkWords}(\lfloor r \cdot n \rfloor)$
$k' \leftarrow \text{NbInsertedWords}(3 \cdot k)$ as [8].
// Creation of the SECTABLE
$\langle \text{SECTABLE} \rangle \leftarrow \Pi_{\text{CSECT}}(wordList, embeddingList)$
// Embed plaintext t with EMBEDDER
$\langle e_t \rangle \leftarrow \Pi_{\text{Embed}}(t,\ \text{EMBEDDER.params})$
// Words Selection
$\langle \text{sims} \rangle \leftarrow \Pi_{\text{CosineSim}}(\langle e_t \rangle,\ \langle \text{SECTABLE} \rangle)$
$\langle W_{\text{candidates}} \rangle \leftarrow \Pi_{\text{TopK}}(\langle \text{sims} \rangle, k')$
$\langle \text{simsFiltered} \rangle \leftarrow \Pi_{\text{CosineSim}}(\langle e_t \rangle, \langle W_{\text{candidates}} \rangle)$
$\langle W_{\text{filtered}} \rangle \leftarrow \Pi_{\text{TopK}}(\langle \text{simsFiltered} \rangle, k)$.
// INSERTER
$P \leftarrow$ "Insert words: W_{filtered} into the text: t"
$\langle t' \rangle \leftarrow \Pi_{\text{Insert}}(P, t)$
$L_{wm} \leftarrow \text{Save}(W_{\text{filtered}})$ ▷ Save watermark words
return Reveal t'

Algorithm 2 Protocol Π_{CSECT} Creation of SECTABLE

Require: Party P_1 holds the vocabulary V. Party P_2 holds the set of embeddings D.
Ensure: Secure embedding table ⟨SECTABLE⟩, and host mappings word2idx, idx2word.

// On Host (e.g., Party P_1's machine)
word2idx, idx2word ← CreateMappings(V) ▷ Create local word-to-index mappings
$M \leftarrow |V|$ ▷ Get vocabulary size
// The following operations are performed in 3PC
$N \leftarrow |D|$ ▷ Get the total number of embeddings from P_2
$seed \leftarrow$ randomSeed and key ← PRNGKey($seed$).
perm ← RandomPermutation(key, N) ▷ Generate a secure permutation of indices $[0, \ldots, N-1]$
chosen_indices ← perm[: M] ▷ Select the first M indices.
⟨SECTABLE⟩ ← D[chosen_indices] ▷ Create the secure table by selecting the corresponding embeddings
return ⟨SECTABLE⟩, word2idx, idx2word

C. Watermark Detection

Following PostMark [8], we need to compute the presence score in watermark detection:

$$p = \frac{|\{w \in \mathbf{w} \text{ s.t. } \exists w' \in t', \text{sim}(w', w) \geq \theta_{sim}\}|}{|\mathbf{w}|}, \quad (2)$$

where $\mathbf{w}$ is the list of watermark words.

Algorithm 3 Protocol Π_{Det} Watermark Detection

Require: Party P_1 holds candidate text t' and watermark word list L_{wm}, party P_2 holds embedded parameters. Similarity threshold $\theta_{sim} = 0.85$ and detection threshold $\theta_{det} = 45\%$ are constants following [8].

Ensure: Boolean indicating if a watermark is detected

Let $W_{cand} \leftarrow$ extract words from t'
$\langle E_{cand} \rangle \leftarrow \Pi_{\text{Embed}}(W_{cand},$ EMBEDDER.params)
$\langle E_{wm} \rangle \leftarrow \Pi_{\text{Embed}}(L_{wm},$ EMBEDDER.params)
$\langle c \rangle \leftarrow 0$
for each word embedding $e_w \in E_{wm}$ **do**
  for each word embedding $e'_w \in E_{cand}$ **do**
    $\langle c \rangle = \langle c \rangle + (\langle \Pi_{\text{CosineSim}}(e_w, e'_w) \rangle \geq \theta_{sim})$
  end for
end for
$\langle p \rangle = \langle c \rangle / |L_{wm}|$. ▷ Calculate the presence score
return $\langle b \rangle = \langle p \rangle > \theta_{det}$

V. Experiments

A. Experimental Setup

Implementation. PrivMark is implemented via SecretFlow-SPU [25] on a 48-CPU server (AMD EPYC 7443P, 256GB RAM) running Ubuntu 22.04 LTS (Linux 5.15.0-82-generic). We use a CPU-only implementation due to the experimental nature of GPU-supported SPU. We evaluate GPT-2 base against a plaintext CPU baseline across three network settings: Localhost = (26Gbps, 0.05ms), LAN = (1.5Gbps, 1.5ms), and WAN = (400Mbps, 10ms).

Models & Datasets. We use `intfloat/e5-base` [26] as our embedding model and variants of GPT-2 [28] as the `INSERTER`. Since we use a different embedding model from PostMark [8], we construct our own vocabulary embedding set D from the WikiText-103 corpus [27].

Availability. We provide a reproducible implementation in https://github.com/CPS4AI/OpenPrivMark.

B. Performance of Secure Watermarking

We evaluate the performance of PrivMark's insertion and detection phases in respective § V-B1 and § V-B2.

1) Performance of Secure Insertion: We benchmarked operations (`Embed`, `Cosine`, `Topk`, `Insert`) on GPT-2 (50 tokens) across Localhost, LAN, and WAN (Table II). Π_{MarkIn} overhead exceeds plaintext significantly. LAN overhead is modest (1.6× for `Embed`, 1.2× for `Insert`) vs. Localhost. However, WAN latency dramatically spikes costs: `Embed`, `Cosine`, and `Topk` slow by 16.5× , 35× , and 123× , respectively. This demonstrates that network latency is a critical factor that magnifies the overhead of secure protocols.

We also measure the overhead of PrivMark in Table II. Similarly to time profiling, the `Insert` operation accounts for more than 90% of the communication overhead.

2) Performance of Secure Detection: We benchmark the performance of Π_{Det} in Table III. It shows that Π_{Det} incurs approximately 145× overhead compared to the 1.17s plaintext baseline. This slowdown aligns with § V-B1, as detection relies on expensive MPC embedding and similarity calculations, alongside network latency.

TABLE II: Performance of Protocol Π_{MarkIn} of PrivMark with GPT-2 base across different networks. All times are in seconds. The `Insert` in WAN was not completed in 1 hour.

Operation	CPU (s)	Π_{MarkIn}			
		Lo (s)	LAN (s)	WAN (s)	Comm. (MB)
`Embed`	1.191	23.76	37.95	393.4	2285
`Cosine`	0.038	0.047	0.060	1.775	0.221
`Topk`	0.006	0.008	0.012	0.986	0.429
`Insert`	57.704	1267	1551	> 1 hour	71431

TABLE III: Performance of Protocol Π_{Det} of PrivMark under different networks.

Operation	CPU (s)	Lo (s)	LAN (s)	WAN (s)	Comm. (MB)
Π_{Det}	1.17	172	321.6	4037	26795

```
[Candidate Words]
Top 6: ['lawyer', 'decade', 'lyrical', 'room', 'print', 'stance']
Filtered Top 2: ['lyrical', 'lawyer']

[PrivMark / Plaintext Insertion Output]
'Insert the following words: [lyrical', 'lawyer'] into the text:
"We are not! Like men in the story of the Good Samaritan, they pass by on the other side..."'
The first sentence of the sentence is a reference to the fact that the Good Samaritan is a man.  The second sentence is a reference to the fact that the Good Samaritan is a woman.
```

Fig. 3: Example of insertion outputs of PrivMark. Plaintext baseline produces the same output.

C. Evaluation of Effectiveness and Robustness

Example of Insertion Outputs. To empirically show that our PrivMark is numerically precise in the watermarking insertion, we first give a representative example to compare our generated outputs to the plaintext CPU baseline. Figure 3 shows semantically identical results, confirming correct logic execution under privacy constraints.

Experimental Evaluation. We evaluate end-to-end effectiveness and robustness using 70 multi-domain examples (e.g., technology, history, astronomy) generated by Gemini 2.5 Pro [29]. The detailed examples are illustrated in our repository. The measure methodology involved a true positive test to measure robustness against attacks and a false positive test to measure accuracy: i) True Positive Test: Checks the original text, paraphrased versions, and partial removal versions. ii) False Positive Test: Checks for watermark signals in unrelated, cross-domain texts.

Table IV shows 100% detection in original text, 91.2% robustness to paraphrasing, and 79.4% to removal, with a 13.2% false positive rate. By design, *Plaintext* (without MPC) and PrivMark report identical metrics, as PrivMark executes the same logic via MPC without degrading detectability or robustness.

VI. Conclusion

We propose PrivMark, the first framework that integrates secure multi-party computation (MPC) with large language

TABLE IV: Detection rates under different attacks.

Test Scenarios	Plaintext	PRIVMARK
Original (True Positive)	100.0%	100.0%
Paraphrase Attack	91.2%	91.2%
Removal Attack	79.4%	79.4%
Unrelated Text (False Positive)	13.2%	13.2%

model (LLM) watermarking. Our work addresses the critical need for content traceability and ownership verification in privacy-sensitive scenarios where models, data, and watermarking keys cannot be exposed. This work serves as a foundational step, demonstrating the feasibility of MPC-based LLM watermarking. The proposed solution utilizes mature cryptographic components and older LLMs, leaving scalability to modern large-scale models for future exploration. Supporting contemporary LLMs requires decomposing modern architectures into MPC-compatible operations, a task facilitated by open-source implementations. Future work can focus on optimizing cryptographic protocols, improving efficiency, and exploring scalability to very large models and settings with many participating parties to enable private, robust, and scalable watermarking for modern LLMs.

ACKNOWLEDGMENTS

This work is supported by the National Research Foundation, Singapore, and Cyber Security Agency of Singapore under its National Cybersecurity R&D Programme and CyberSG R&D Cyber Research Programme Office. Any opinions, findings and conclusions or recommendations expressed in these materials are those of the author(s) and do not reflect the views of National Research Foundation, Singapore, Cyber Security Agency of Singapore as well as CyberSG R&D Programme Office, Singapore.

REFERENCES

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[2] G. Kendall and J. A. Teixeira da Silva, "Risks of abuse of large language models, like chatgpt, in scientific publishing: authorship, predatory publishing, and paper mills." *Learned Publishing*, vol. 37, no. 1, 2024.

[3] L. Pan, A. Liu, Z. He, Z. Gao, X. Zhao, Y. Lu, B. Zhou, S. Liu, X. Hu, L. Wen *et al.*, "Markllm: An open-source toolkit for llm watermarking," *arXiv preprint arXiv:2405.10051*, 2024.

[4] Y. Liang, J. Xiao, W. Gan, and P. S. Yu, "Watermarking techniques for large language models: A survey," *arXiv preprint arXiv:2409.00089*, 2024.

[5] S. Abdelnabi and M. Fritz, "Adversarial watermarking transformer: Towards tracing text provenance with data hiding," 2021. [Online]. Available: https://arxiv.org/abs/2009.03015

[6] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein, "A watermark for large language models," 2024. [Online]. Available: https://arxiv.org/abs/2301.10226

[7] X. Zhao, P. Ananth, L. Li, and Y.-X. Wang, "Provable robust watermarking for ai-generated text," *arXiv preprint arXiv:2306.17439*, 2023.

[8] Y. Chang, K. Krishna, A. Houmansadr, J. F. Wieting, and M. Iyyer, "PostMark: A robust blackbox watermark for large language models," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 8969–8987. [Online]. Available: https://aclanthology.org/2024.emnlp-main.506/

[9] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 1986, pp. 162–167.

[10] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," pp. 218–229, 01 1987.

[11] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. IEEE, 2001, pp. 136–145.

[12] P. Mohassel and P. Rindal, "Aby3: A mixed protocol framework for machine learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2018, p. 35–52. [Online]. Available: https://doi.org/10.1145/3243734.3243760

[13] Y. Liang, J. Xiao, W. Gan, and P. S. Yu, "Watermarking techniques for large language models: A survey," 2024. [Online]. Available: https://arxiv.org/abs/2409.00089

[14] R. Sato, Y. Takezawa, H. Bao, K. Niwa, and M. Yamada, "Embarrassingly simple text watermarks," 2023. [Online]. Available: https://arxiv.org/abs/2310.08920

[15] X. Yang, K. Chen, W. Zhang, C. Liu, Y. Qi, J. Zhang, H. Fang, and N. Yu, "Watermarking text generated by black-box language models," 2023. [Online]. Available: https://arxiv.org/abs/2305.08883

[16] T. Munyer, A. Tanvir, A. Das, and X. Zhong, "Deeptextmark: A deep learning-driven text watermarking approach for identifying large language model generated text," 2024. [Online]. Available: https://arxiv.org/abs/2305.05773

[17] R. Zhang, S. S. Hussain, P. Neekhara, and F. Koushanfar, "REMARK-LLM: A robust and efficient watermarking framework for generative large language models," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 1813–1830. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/zhang-ruisi

[18] D. Li, R. Shao, H. Wang, H. Guo, E. P. Xing, and H. Zhang, "Mpcformer: fast, performant and private transformer inference with mpc," 2023. [Online]. Available: https://arxiv.org/abs/2211.01452

[19] M. Hao, H. Li, H. Chen, P. Xing, G. Xu, and T. Zhang, "Iron: Private inference on transformers," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=deyqjpcTfsG

[20] Y. Dong, W. jie Lu, Y. Zheng, H. Wu, D. Zhao, J. Tan, Z. Huang, C. Hong, T. Wei, and W. Chen, "Puma: Secure inference of llama-7b in five minutes," 2023. [Online]. Available: https://arxiv.org/abs/2307.12533

[21] W.-j. Lu, Z. Huang, Z. Gu, J. Li, J. Liu, K. Ren, C. Hong, T. Wei, and W. Chen, "Bumblebee: Secure two-party inference framework for large transformers," *Cryptology ePrint Archive*, 2023.

[22] Q. Pang, J. Zhu, H. Möllering, W. Zheng, and T. Schneider, "Bolt: Privacy-preserving, accurate and efficient inference for transformers," *Cryptology ePrint Archive*, 2023.

[23] K. Gupta, N. Jawalkar, A. Mukherjee, N. Chandran, D. Gupta, A. Panwar, and R. Sharma, "Sigma: secure gpt inference with function secret sharing," *Cryptology ePrint Archive*, 2023.

[24] P. Mohassel and P. Rindal, "ABY3: A Mixed Protocol Framework for Machine Learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. ACM, 2018, pp. 35–52.

[25] J. Ma, Y. Zheng, J. Feng, D. Zhao, H. Wu, W. Fang, J. Tan, C. Yu, B. Zhang, and L. Wang, "SecretFlow-SPU: A performant and User-Friendly framework for Privacy-Preserving machine learning," in *2023 USENIX Annual Technical Conference (USENIX ATC 23)*. Boston, MA: USENIX Association, Jul. 2023, pp. 17–33.

[26] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, and F. Wei, "Text embeddings by weakly-supervised contrastive pre-training," *arXiv preprint arXiv:2212.03533*, 2022.

[27] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," 2016. [Online]. Available: https://arxiv.org/abs/1609.07843

[28] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, 2019.

[29] G. Comanici, E. Bieber, M. Schaekermann, and et al., "Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities," 2025. [Online]. Available: https://arxiv.org/abs/2507.06261