NSMARK: NULL SPACE BASED BLACK-BOX WATERMARKING DEFENSE FRAMEWORK FOR LANGUAGE MODELS

Haodong Zhao[♠◇] **Jinming Hu**[♠] **Peixuan Li**[♠] **Fangqi Li**[♠] **Jinrui Sha**[♠]
Tianjie Ju[♠] **Peixuan Chen**[◇] **Zhuosheng Zhang**[*♠] **Gongshen Liu**[*♠]
[♠]Shanghai Jiao Tong University [◇]Tencent

ABSTRACT

Language models (LMs) have emerged as critical intellectual property (IP) assets that necessitate protection. Although various watermarking strategies have been proposed, they remain vulnerable to Linear Functionality Equivalence Attack (LFEA), which can invalidate most existing white-box watermarks without prior knowledge of the watermarking scheme or training data. This paper analyzes and extends the attack scenarios of LFEA to the commonly employed black-box settings for LMs by considering Last-Layer outputs (dubbed LL-LFEA). We discover that the null space of the output matrix remains invariant against LL-LFEA attacks. Based on this finding, we propose NSMARK, a black-box watermarking scheme that is task-agnostic and capable of resisting LL-LFEA attacks. NSMARK consists of three phases: (i) watermark generation using the digital signature of the owner, enhanced by spread spectrum modulation for increased robustness; (ii) watermark embedding through an output mapping extractor that preserves the LM performance while maximizing watermark capacity; (iii) watermark verification, assessed by extraction rate and null space conformity. Extensive experiments on both pre-training and downstream tasks confirm the effectiveness, scalability, reliability, fidelity, and robustness of our approach. Code is available at https://github.com/dongdongzhaoUP/NSmark.

1 INTRODUCTION

Over the past few decades, language models (LMs) have achieved exceptional performance and found applications across a wide range of fields. However, training high-performance LMs requires vast amounts of data and significant computational resources, making these models valuable intellectual property (IP). With the rise of machine learning as a service (MLaaS) platforms, companies sell well-trained LMs as commodities and release APIs for public access. Once these models are illegally stolen, distributed or resold, the rights of the model owners are severely violated. Therefore, protecting the intellectual property of LMs is essential.

Watermarking techniques have been widely used to protect the IP of deep learning models Chen et al. (2024); He et al. (2024); Carlini et al. (2024); Feng et al. (2024). By incorporating identifiable information, these techniques could verify model ownership and provide proof of authenticity. Existing watermarking schemes can be categorized into white-box and black-box approaches, depending on whether the model parameters need to be accessed in verification. Among these, black-box schemes are more applicable in real-life scenarios, where model parameters are often inaccessible, such as in cases where models are deployed as APIs.

However, protecting the IP of LMs through watermarking presents significant challenges. Since LMs can be deployed for various post-training downstream tasks Zhang et al. (2023), it is crucial that watermark schemes remain task independent. Furthermore, recent studies have revealed vulnerabilities and shortcomings in existing watermarking techniques Li et al. (2023a). Specifically, the proposed Linear Functionality Equivalence Attack (LFEA) is simple to conduct and can compromise most existing white-box watermarks by exploiting linear invariance without knowledge of the watermarking

[*]Corresponding authors.

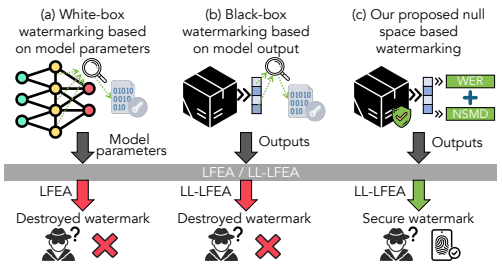

Figure 1: Illustration of different watermark schemes against LFEA/LL-LFEA. LFEA disables parameters based white-box schemes Li et al. (2023a) and LL-LFEA disables output based black-box schemes (Section 3.1). NSMARK is secure against LL-LFEA using null space invariance.

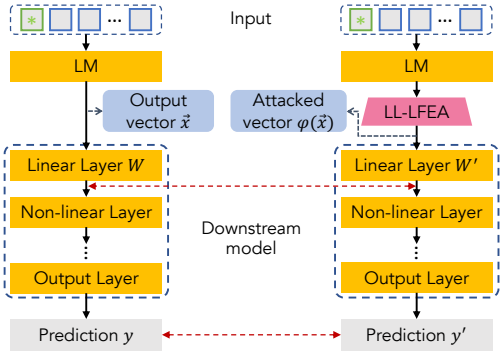

Figure 2: The schematic diagram of model inference flow before and after LL-LFEA attack. LL-LFEA transforms the LM output and performs an inverse transform in the subsequent linear layer, leaving the final prediction unchanged.

scheme or the training data. As the hidden states and outputs of the last layer in LM are widely used for classification and generation tasks, we consider them, analyze and expand LFEA scenarios to black-box settings utilizing model outputs (dubbed LL-LFEA).

In this work, we first explore the characteristics of the model output. We discover that the null space of the matrix composed of the model output vectors is invariant under LL-LFEA. Based on this finding, we propose a new null space verification method that can withstand the LL-LFEA attack. This method uses a new metric, the Null Space Matching Degree (NSMD). NSMD measures the degree of match between the output matrix of the suspicious model and the null space of the protected LM. Finally, we propose NSMARK, a null-space-based task-agnostic black-box watermarking scheme for LMs. NSMARK uses identity information to generate all elements related to the watermark and uses the Watermark Extracting Rate (WER) and NSMD to verify the watermark, thus can pass through as shown in Figure 1. Spread spectrum modulation technology and an extra extractor are also introduced to enhance watermark performance.

Our contributions are summarized as follows:

(i) We analyze the threat of LFEA on output-based watermark and propose LL-LFEA, which can destroy the watermark embedded in the output vector without affecting the performance of downstream tasks.

(ii) We find that the null space of the matrix composed of the output vectors of the model is invariant under LL-LFEA and thus propose a new null space verification method NSMARK which can resist LL-LFEA. Notably, NSMARK is task-agnostic that uses both new null space verification and signature verification to resist LL-LFEA.

(iii) We conduct comprehensive experiments by applying NSMARK to various models of pre-training and downstream tasks. The experimental results demonstrate the effectiveness, fidelity, reliability, and robustness of NSMARK.

2 Related Work

Watermarking for LMs. With the rise of pre-training in NLP, recent work has explored watermarking specific to LMs. BadPre Jia et al. (2022) introduced a task-agnostic backdoor attack only for MLM-based LMs. Hufu Xu et al. (2024) introduced a modality-agnostic approach for pre-trained Transformer models using the permutation equivariance property. Explanation as a Watermark Shao et al. (2024) addressed the limitations of backdoor-based techniques by embedding multi-bit watermarks into feature attributions using explainable AI. Peng et al. (2023); Shetty et al. (2024) proposed Embeddings-as-a-Service (EaaS) watermarks to protect the intellectual property of EaaS providers. Shen et al. (2021); Zhang et al. (2023) proposed task-agnostic backdoor attacks by assigning high-dimensional vectors as trigger set labels, but their effectiveness is sensitive to downstream classifier initialization. Wang & Kerschbaum (2021) introduced an auxiliary neural network for watermark embedding using weights from the main network. Wu et al. (2022) proposed a task-agnostic

embedding loss function, but didn't consider the need for triggers to reflect the model owner's identity. Cong et al. (2022) introduced a black-box watermarking scheme for PLMs, but its applicability is limited due to the discrete nature of word tokens. Unfortunately, these schemes are vulnerable to attacks by LFEA or LL-LFEA in principle.

Watermark Removal Attacks. DNN watermarking faces various removal attempts. Common methods include fine-tuning Adi et al. (2018) and pruning Han et al. (2015). Fine-pruning Liu et al. (2018) combines these approaches for greater effectiveness. Knowledge Distillation Hinton (2015) techniques can also inadvertently remove watermarks while reducing model size. Shetty et al. (2024) show that existing EaaS watermarks can be removed by paraphrasing attack. Lukas et al. (2022) propose a new attack method called Neuron Reordering to swap neurons within the same hidden layer of a DNN to disrupt embedded watermarks in the model's parameters. Li et al. (2023a) introduce a powerful LFEA for white-box watermarks, applying linear transformations to model parameters, effectively destroying embedded watermarks while preserving the model's original functionality. Fraud attacks include overwriting Wang & Kerschbaum (2019) and ambiguity attacks Zhu et al. (2020) also pose a great threat to watermarks.

3 METHOD

3.1 THREAT MODEL

In white-box watermarking schemes, high-dimensional model parameters are often used as watermark information. For LMs, since the output of the last layer is also high-dimensional, we can use a method similar to the white-box schemes to embed watermarks in the output. However, embedding identity information into the high-dimensional output vector will face the threat of LFEA-like attacks, which is proposed to destroy watermark information embedded in model parameters by linearly transforming parameters of intermediate layers. Next, we discuss the specific form of linear isomorphism attacks.

Assume the attacker knows the watermark information is embedded in the LM output and seeks to remove the watermark with minimal attack cost (without modifying the model structure or fine-tuning the model) while ensuring that the model's normal task performance remains unaffected. As shown in Figure 2, we propose an attack method that satisfies this requirement and provide a proof below.

The output vector $\vec{x}$ is generated by the LM and serves as input to the downstream model. After passing through a series of linear and non-linear layers, the prediction result y is obtained. The attacker attempts to modify the output vector of the LM to destroy the watermark while ensuring that the final prediction remains unaffected. Specifically, the attacker changes $\vec{x}$ to $\varphi(x)$ and inputs it into the downstream model, so that the resulting prediction y' remains equal to the original prediction y.

The sufficient condition for this result is that the modification to the LM output vector is compensated for after passing through the first linear layer of the downstream network. Let the parameter matrix of the first linear layer in the downstream network be denoted by W. In this case, the attacker aims to satisfy the following condition: $W'\varphi(\vec{x}) = W\vec{x}$, which leads to $\varphi(\vec{x}) = W'^{,\dagger}W\vec{x} = Q\vec{x}$, where $Q = W'^{,\dagger}W$ and $W'^{,\dagger}$ is the pseudo-inverse of W' Li et al. (2023a). To avoid loss of information during the linear transformation (since this would adversely affect downstream tasks), Q must be a reversible matrix. We present a simple method and analysis on how to quickly generate high-dimensional Q in Appendix A.1.

We show that the attacker can apply a linear transformation to $\vec{x}$ thereby destroying the watermark embedded in the output vector, while leaving the downstream task performance unchanged. We refer to this attack as the Last-Layer Linear Functionality Equivalence Attack (LL-LFEA). In addition to the theoretical analysis, the effectiveness of LL-LFEA is experimentally verified in Appendix B.1.

3.2 NULL SPACE VERIFICATION THEORY

LL-LFEA applies a linear transformation to the output vector of LM and can destroy the embedded watermark. As a result, previous watermark verification methods may be significantly impacted. We observe that the null space of the matrix composed of the output vector is invariant under the LL-LFEA attack. Based on this, we propose to use the null space matching degree to verify whether the model is embedded with watermarks.

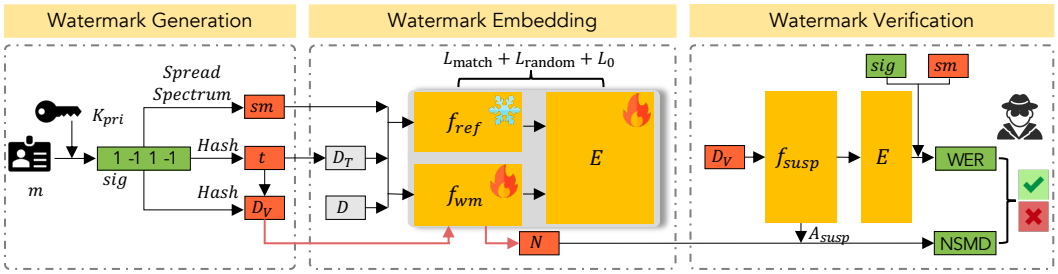

Figure 3: The overall workflow of NSMARK. (i) In watermark generation, identity information is used generate sig. (ii) In watermark embedding, watermarked model f_{wm} and extractor E are trained with the participation of the reference model f_{ref}. (iii) In watermark verification, WER and NSMD collaborate to verify the identity of the model.

Theorem 1. *Before and after LL-LFEA, the null space of the output matrix of PLM remains unchanged for the same input set. (Proofs in Appendix A.2)*

Therefore, even if the watermark based on the digital string is corrupted, we can still verify model ownership using the null space of the output matrix.

3.3 NULL SPACE MATCH DEGREE (NSMD)

We define NSMD by introducing the distribution of elements in a matrix, which is obtained by multiplication of the matrix of the output matrix A of any LM without watermark and the null space matrix N of f_{wm}. In $H_{(n\times p)} = A_{(n\times m)} \times N_{(m\times p)}$, $H_{i,j} = \alpha_i \cdot \beta_j$ is the dot product of the ith row vector of A and the jth column vector of N. We define NSMD of A and N as:

$$\text{NSMD}(A, N) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{p} \sqrt{|H_{i,j}|}. \tag{1}$$

Furthermore, we give a detailed analysis of estimation of NSMD (in Appendix A.3). For example, if $n = 768$ and $p = 1500$, we have NSMD > 27.48. If N is the null space matrix of A, NSMD is a minimum value close to 0. This difference is amplified by the process of calculating the square root, resulting in a significant difference between whether A and N are matched. We use this difference to distinguish whether the model is embedded with a watermark.

3.4 OVERALL FRAMEWORK OF NSMARK

NSMARK includes three modules: watermark generation, watermark embedding, and watermark verification, as shown in Figure 3. We describe the modules as follows.

3.4.1 WATERMARK GENERATION

Algorithm 1 shows the watermark generation workflow. We hope that the generated watermark contains the owner's identity information. First, the digital signature $sig = \textbf{Sign}(m)$ is generated from the identity information message m. To ensure that the trigger t has a unique mapping relationship with sig, only one trigger is used. We use the trigger generation algorithm $\textbf{Encode}(\cdot)$ introduced in Li et al. (2023b) to obtain $t = \textbf{Encode}(sig, n = 1)$. t is inserted into clean sample x of dataset D to form a trigger set D_T.

To defend against ambiguous attacks, the verification trigger set D_V used for the null space verification also needs to be generated based on sig. A candidate pool D_{NS} to generate null space verification data sets should be published, and then a fixed number of samples are selected from the D_{NS} based on the digital signature as the verification data set D_V. We define the verification data set selection algorithm as $\textbf{Select}(sig) \rightarrow D_V$, which must be a deterministic algorithm, that is, for the same input, there must be the same output. In addition, we hope that the algorithm will have different outputs for

different inputs. Therefore, we choose a hash function and use a one-way hash chain to generate D_V. We hope that the index repetition rate obtained by different hash-value mappings is low, so we hope that the data set D_{NS} is as large as possible. The specific process of the **Select**$(\cdot)$ algorithm is shown in Algorithm 2.

To improve the robustness of the watermark, we introduce the spread spectrum modulation technology as Feng & Zhang (2020). Spread spectrum modulation technology uses redundant bits to represent the original information. Figure 7 shows an example of the spreading of $3\times$. Please refer to Appendix A.6.1 for the specific process $\mathbf{SM}(sig) \rightarrow sig_{wm}$.

3.4.2 WATERMARK EMBEDDING

Before the training starts, make a copy of f_{wm} as the frozen reference model f_{ref}. Then use the clean data set D and the trigger set D_T to train f_{wm} and the extractor E. When taking $\{D, D_T\}$ as input, f_{wm} will output $\{V, V^T\}$ and f_{ref} will output $\{V_{ref}, V_{ref}^T\}$, respectively. For V^T, E maps it to obtain the signature sig_{wm}, and for $\{V, V_{ref}, V_{ref}^T\}$, E maps them to random vectors. After the training is completed, f_{wm} is embedded with the watermark. Then using D_V as input, the output vectors are concatenated into a matrix A, and the corresponding null space matrix N of A is calculated as part of the key.

Three networks are involved in watermark embedding: the model f_{wm} to be embedded with the watermark, the reference model f_{ref} and the extractor model E. Compared to directly embedding sig in the output vector of f_{wm}, adding E to the map can reduce the side effect of the watermark on the original performance. The watermark capacity is increased at the same time. We use the mean square error loss (MSE) and the similarity function sim to implement the above training process:

$$L_{\text{match}} = \frac{1}{|D_T|} \sum_{x \in D_T} \mathrm{MSE}\left(E\left(V^T\right), sig_{sm}\right), \tag{2}$$

$$L_{\text{random}} = \frac{1}{|D|} \sum_{x \in D} \mathrm{sim}\left(E\left(V\right), sig_{sm}\right)^2 + \frac{1}{|D_T|} \sum_{x \in D_T} \mathrm{sim}\left(E\left(V_{ref}^T\right), sig_{sm}\right)^2 + \frac{1}{|D|} \sum_{x \in D} \mathrm{sim}\left(E\left(V_{ref}\right), sig_{sm}\right)^2. \tag{3}$$

We use cosine similarity as the sim function. The complete loss function of E is $L_{\text{Extractor}} = \lambda_1 L_{\text{match}} + (1-\lambda_1) L_{\text{random}}$. During training, only the parameters of E are trainable. The loss of f_{wm} also consists of two parts: $L_{wm} = \lambda_2 L_{match} + (1-\lambda_2) L_0$. The content of this L_{match} is the same as L_{match} of E, but only the parameters of f_{wm} are updated at this time, and L_0 is the original LM training loss function. During training, E and f_{wm} are trained alternately.

3.4.3 WATERMARK VERIFICATION

To effectively defend attacks, NSMARK uses two metrics together to verify ownership: WER and NSMD. Model owner needs to submit $key = (sig, E, N)$ to the Certification Authority (CA). CA generates t, D_V, sm using sig. Input D_V to the suspicious model f_{susp} to get the output vector A_{susp}, and pass A_{susp} through E to get the mapped vector O_{susp}. Then WER is obtained from the despread spectrum. WER is defined from comparing bits in sig and sig':

$$\text{WER} = \frac{1}{n} \sum_{i=0}^{n-1} \left[a_i = a_i'\right], \tag{4}$$

where $[\cdot]$ is the *inverse bracket*, which is 1 when the expression in the bracket is *True*, otherwise 0.

NSMD is calculated using A_{susp} and N by Equation 1. We define two thresholds, and whether WER$> T_W$ will be first verified. If it fails, whether NSMD$< T_N$ will be further considered in the case of LL-LFEA.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets. We use WikiText-2 Merity et al. (2017) for pre-training and watermark embedding. To evaluate the performance on downstream tasks, we select many text classification datasets: SST-2 and

SST-5 Socher et al. (2013) for sentiment analysis, Lingspam Sakkis et al. (2003) for spam detection, OffensEval Zampieri et al. (2019) for offensive language identification and AG News Zhang et al. (2015) for news classification.

Table 1: Effectiveness of NSMARK on different LMs. f_{wm} means watermarked model and f_{clean} is not watermarked. WER=1.00 indicates that the signature information can be accurately extracted, and NSMD has obvious differentiation between f_{wm} and f_{clean}.

Metric	f_{wm}				f_{clean}			
	BERT	RoBERTa	DeBERTa	XLNet	BERT	RoBERTa	DeBERTa	XLNet
WER	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.03
NSMD	2.94×10^{-6}	2.53×10^{-6}	2.91×10^{-6}	2.90×10^{-6}	60.95	61.24	87.88	76.74

Models. For LMs, we use the base versions of BERT Kenton & Toutanova (2019), RoBERTa Liu (2019), DeBERTa He et al. (2020) and XLNet Yang (2019) for main results. Llama-2-7B Touvron et al. (2023), GPT-2 Radford et al. (2019) and the large version of BERT and RoBERTa are also used in supplementary experiments. All pretrained weights are from HuggingFace.[1] The extractor network is a three-layer linear network with hidden layers of neurons 2048 and 1024. The input dimension matches the output dimension of the LM. The output dimension matches the size of sig_{sm}.

Watermark settings and training details. We select a string containing owner information as the message m, for example, "BERT is proposed by Google in 2018". The length of sig is 256 and then spread spectrum by a factor $k = 3$, resulting in a 768-bit sig_{sm}. SST-2 is used as the candidate pool D_{NS}. q, the length of D_V, is 1500. The trigger is inserted into random positions for 5 times in the trigger set. When performing watermark embedding, $\lambda_1 = 0.5$ and $\lambda_2 = 0.2$ in $L_{Extractor}$ and L_{wm}. The batchsize is 4, and the learning rates for both f_{wm} and E are 10^{-4}. f_{wm} and E are trained alternately for the 10 epochs. When fine-tuning downstream tasks, the learning rate is 2×10^{-5} and the batchsize is 8 for 3 epochs.

Metrics. As mentioned before, two metrics are defined to verify the identity of the model: WER and NSMD. Besides, we adopt accuracy (ACC) to measure the performance of LM on downstream tasks.

4.2 LL-LFEA ATTACK EVALUATION

We select the effective *word embedding-based watermarking scheme (EmbMarker)* Peng et al. (2023) and NSMARK (without NSMD) as victims to study the effectiveness of LL-LFEA. The attack results on EmbMarker are shown in Appendix B.1 and the results on NSMARK (without NSMD) are shown in Table 4. Though EmbMarker can pass through the attack of RedAlarm Zhang et al. (2023), after the LL-LFEA attack, all the metrics of EmbMarker are very close to those of the original model without watermark. At the same time, LL-LFEA has little degradation on original model performance. This fully demonstrates the effectiveness of LL-LFEA on existing watermarking schemes.

4.3 NSMARK PERFORMANCE EVALUATION

We analyze the main experimental results about NSMARK. For more results on computational cost and other more detailed experiments, please refer to Appendix B.

4.3.1 EFFECTIVENESS

Effectiveness means that the watermark can achieve the expected effect during verification. Ideally, sig' extracted from the watermarked model should be consistent with the original sig, and the output matrix of f_{wm} for D_V should match completely N stored in key, which means WER = 1 and NSMD = 0. Table 1 shows the results of f_{wm} embedded with watermark and f_{clean} without watermark. It can be seen that for different watermarked LMs, WER is 1 and NSMD is close to 0. This shows the effectiveness of NSMARK. Comparison of the values of f_{wm} and f_{clean} shows that WER and NSMD will obviously change after the watermark is embedded. Although NSMD of different LMs has different values, they are all far from 0. Through these results, we can preliminarily define

[1] https://huggingface.co/

Table 2: WER results of different extractor E. f_{wm}, E_c, and E_w means watermarked model, correct E and wrong E, respectively. Only the correct E can accurately extract the signature.

Setting	BERT	RoBERTa	DeBERTa	XLNet
$f_{wm} + E_c$	1.00	1.00	1.00	1.00
$f_{wm} + E_w$	0.00	0.00	0.00	0.13

Table 3: NSMD of different null space matrix N. f_{wm}, N_c, N_r and N_s means watermarked model, correct N, random N and N composed of small elements, respectively. Only when the correct N is used can the NSMD be close to 0.

Setting	BERT	RoBERTa	DeBERTa	XLNet
$f_{wm} + N_c$	2.94×10^{-6}	2.53×10^{-6}	2.91×10^{-6}	2.90×10^{-6}
$f_{wm} + N_r$	3167.81	3171.58	3182.79	3117.61
$f_{wm} + N_s$	1001.75	1002.94	1006.49	985.87

verification thresholds of WER and NSMD as $T_W = 0.6$ and $T_N = 43$, which are $0.6\times$ the average gaps. Thresholds can be further adjusted according to different models and task types. We also study the watermark effectiveness on larger size of the models in Table 10, which demonstrates the scalability of NSMARK.

4.3.2 RELIABILITY

The watermark key is a triple $key = (sig, E, N)$. Next, we analyze whether the watermark can be successfully verified if an attacker provides an incorrect key.

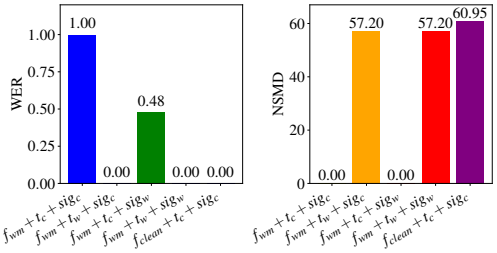

Figure 4: The impact of the correctness of trigger t and signature sig on WER and NSMD. The c in the subscript stands for *correct* and w stands for *wrong*. Only f_{wm} with correct trigger and sig can pass through verification.

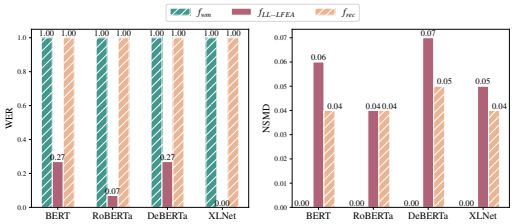

Figure 5: Changes of WER and NSMD before and after LL-LFEA attack and recovery. f_{wm}, $f_{LL-LFEA}$ and f_{rec} denote watermarked model, f_{wm} attacked by LL-LFEA, recovered model from $f_{LL-LFEA}$, respectively.

Wrong signature sig**.** The trigger t and the output of f_{wm} are related to sig, but as sig and t are not a one-to-one mapping relationship, there are situations where only one of sig and t is correct. Figure 4 shows all possible scenarios.

For f_{wm}, (i) when the trigger is wrong (t_w) and the signature is correct (sig_c), WER = 0, NSMD $> T_N$. This means that f_{wm} has learned the relationship between sig and t. Whether t is correct determines whether f_{wm} can produce the expected output, which in turn affects both the calculation of WER and NSMD. (ii) When the trigger is correct (t_c) and the signature is wrong (considering the most dangerous scenario that sig_w consists only of $\{-1, 1\}$), WER ≈ 0.5. This is because t_c leads to the right sig', and its expectation of WER with a random string $\{-1, 1\}$ is 0.5. As the output matrix is correctly generated by f_{wm} based on t_c, NSMD = 0 in this case. (iii) When both trigger and signature are wrong (t_w and sig_w), WER = 0, NSMD$> T_N$, indicating that the watermark cannot be correctly verified without providing the correct key. (iv) For a model without embedded watermarks f_{clean}, watermarks cannot be extracted even if the correct key is provided.

Wrong extractor E**.** Since E is not involved in NSMD calculation, we only analyze the impact of E on WER. As shown in Table 2, when E is wrong, the WER is close to 0, indicating that wrong E is unable to extract the watermark.

Wrong null space N**.** Since N is not involved in calculating WER, we only analyze its impact on NSMD. As shown in Table 3, N_r is a randomly generated matrix with the same dimension as N and each element is distributed between $[0, 1]$. It can be seen that NSMD is very large at this time. However, if the attacker knows the watermark algorithm and want to reduce NSMD, a N_s with extremely small elements might be generated. In this case, NSMD might meet the verification requirements. This indicates that NSMD cannot be used independently to verify the watermark.

Table 4: Impact of fine-tuning on watermark performance. F_{wm} means fine-tuned whole watermarked model and F_{clean} denotes fine-tuned model without watermark. (i) F_{wm} has a slight loss in ACC compared to F_{clean}. (ii) Fine-tuning has little effect on the WER of F_{wm}. (3) Fine-tuning increases the NSMD, but it is still significantly different from the model without watermark.

Metric	Model Setting		SST-2	SST-5	Offenseval	Lingspam	AGnews
ACC	BERT	F_{wm}	91.40	52.62	85.12	99.14	93.95
		F_{clean}	91.63	53.03	84.07	99.66	94.38
	RoBERTa	F_{wm}	92.55	54.71	84.30	99.66	94.43
		F_{clean}	94.04	56.15	84.88	100.00	94.72
	DeBERTa	F_{wm}	93.00	55.48	83.02	99.31	94.61
		F_{clean}	93.58	57.65	85.12	99.31	94.84
	XLNet	F_{wm}	88.65	42.67	81.98	99.14	93.29
		F_{clean}	93.58	53.62	84.65	99.31	94.07
WER	BERT	F_{wm}	1.00	1.00	1.00	0.94	1.00
		F_{clean}	0.00	0.00	0.00	0.00	0.00
	RoBERTa	F_{wm}	0.98	1.00	1.00	0.72	0.99
		F_{clean}	0.00	0.00	0.00	0.00	0.00
	DeBERTa	F_{wm}	1.00	1.00	1.00	0.80	0.88
		F_{clean}	0.00	0.00	0.00	0.00	0.00
	XLNet	F_{wm}	1.00	1.00	1.00	0.92	1.00
		F_{clean}	0.00	0.00	0.00	0.01	0.00
NSMD	BERT	F_{wm}	29.77	25.29	22.52	21.96	24.37
		F_{clean}	72.97	70.06	66.65	69.90	61.59
	RoBERTa	F_{wm}	50.17	30.97	25.15	26.78	28.43
		F_{clean}	74.75	74.48	69.90	65.06	75.54
	DeBERTa	F_{wm}	31.89	25.74	23.52	27.23	37.68
		F_{clean}	80.81	76.72	72.41	68.49	76.33
	XLNet	F_{wm}	24.12	23.52	25.29	24.06	26.20
		F_{clean}	76.30	74.75	69.84	78.09	74.97

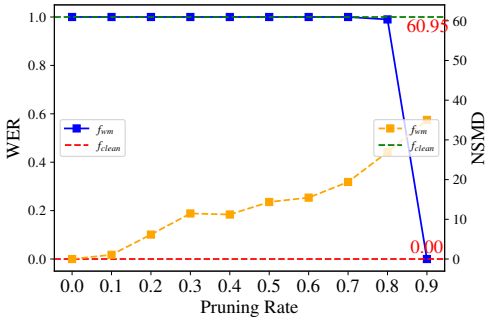

Table 5: Impact of pruning attacks on watermark. The dotted line is the performance of the original model without watermark.

Table 6: Impact of overwriting attacks on watermark performance. f_{ow} is the overwritten model, which is fine-tuned to obtain F_{ow}. "–" means not applicable.

Model	Downstream dataset	ACC	WER	NSMD
f_{ow}	–	–	1.00	22.76
F_{ow}	SST-2	92.32	0.98	48.77
	SST-5	50.54	1.00	36.47
	Offenseval	84.77	1.00	36.61
	Lingspam	99.31	0.62	28.87
	AGnews	93.51	1.00	32.49

Table 7: LL-LFEA results on NSMARK. f_{wm} means watermarked model and $f_{LL-LFEA}$ denotes f_{wm} attacked by LL-LFEA. Due to the invariance of the null space to linear transformations, NSMD can still effectively prove the IP of the model despite the failure of WER.

Metric	BERT		RoBERTa		DeBERTa		XLNet	
	f_{wm}	$f_{LL-LFEA}$	f_{wm}	$f_{LL-LFEA}$	f_{wm}	$f_{LL-LFEA}$	f_{wm}	$f_{LL-LFEA}$
WER	1.00	0.27	1.00	0.07	1.00	0.27	1.00	0.00
NSMD	2.94×10^{-6}	0.06	2.53×10^{-6}	0.04	2.91×10^{-6}	0.07	2.90×10^{-6}	0.05

4.3.3 FIDELITY

We hope that NSMARK does not affect the performance on the original tasks. Thus, we add a downstream network to f_{wm}, and fine-tune the whole model F_{wm} with the downstream dataset. F_{clean} without watermark is fine-tuned as baseline. Table 4 shows that the watermark has almost no impact on the performance of the model on the original task.

4.3.4 DEFENSE AGAINST LL-LFEA

Defense against LL-LFEA. When designing NSMARK, we focus on resisting LL-LFEA and propose null space verification using NSMD. Table 7 shows the impact of LL-LFEA on watermark verification, where $f_{LL-LFEA}$ denotes the model f_{wm} attacked by LL-LFEA. Experiments show that after LL-LFEA, WER drops significantly, as discussed in Section 3.1, but NSMD is still close to 0, verifying that NSMD is an effective indicator for LL-LFEA. Furthermore, after applying LL-LFEA, the attacker can add a network to $f_{LL-LFEA}$ and fine-tune it for downstream tasks (detailed results are presented in Appendix B.3). Additionally, since LL-LFEA causes minimal degradation in model performance, the attacker may attempt to further compromise the watermark through multiple LL-LFEA attacks. Analysis for this aspect is provided in Appendix B.7.

Recovery of WER. In LFEA Li et al. (2023a), a method is proposed to recover the watermark. We revise this method to recover f_{rec} from $f_{LL-LFEA}$. Specifically, assume that the output matrix of f_{wm} is $A_{1(n\times m)}$ as Proof A.2. After being attacked with $Q_{(n\times n)}$, the output matrix turns to

$A_2 = Q \times A_1$. Therefore, an estimate of Q can be obtained as $Q' = A_2 \times A_1^{-1}$. If $m \neq n$, then A_1 is not reversible and $Q' = A_2 \times A_1^T \times (A_1 \times A_1^T)^{-1}$. Then we perform an anti-attack transformation on $f_{LL-LFEA}$, that is, multiply all the outputs of $f_{LL-LFEA}$ by Q' to get f_{rec}. Figure 5 shows that after recovery, WER is significantly improved, indicating that such linear attacks are recoverable. In all cases, the NSMD is quite small, proving that NSMD is invariant to LL-LFEA. In the recovery algorithm in Li et al. (2023a), both the attacker and the model owner might use such an algorithm to claim to be the owner of the model, which will cause verification ambiguity. However, our proposed NSMD is invariant under LL-LFEA, so as long as the timestamp information is added to the key tuple, the ownership can be reliably verified according to the time sequence of the model and the release of key.

4.3.5 ROBUSTNESS

The robustness of watermark refers to whether watermark can be effectively verified after watermark removal attacks. Next, we will analyze the robustness of NSMARK against fine-tuning, pruning, fine-pruning, and overwriting attacks. More robustness analysis against paraphrasing attack and multi-time LL-LFEA attack are shown in Appendix B.6- B.7.

Robustness against fine-tuning. Table 4 shows the WER and NSMD results after fine-tuning on downstream tasks. F_{wm} and F_{clean} are obtained the same as in Section 4.3.3. In most cases, the WER is still very high, indicating that the embedded sig can still be effectively extracted after downstream fine-tuning. However, the WER of RoBERTa on Lingspam task is relatively low. In main results we set the max input length to 128, which is quite shorter than the average length of Lingspam samples (average length of 695.26). Thus it is not sure the model's input includes triggers (possibly truncated). We perform further experiments on increasing the max length of input to 512, and modify the position of trigger to the front. WER increases to more than 0.84 and 0.87 respectively. Besides, compared to F_{clean}, there is still obvious discrimination. Therefore, for complex tasks, the verification threshold T_W can be slightly lowered.

Robustness against pruning and fine-pruning. Pruning is a commonly used model compression method and is often used to destroy the watermark embedded in the model. Referring to Han et al. (2015); Shao et al. (2024), we sort the parameters of each layer in LM, then set some fractions of parameters with smaller absolute value to 0. Figure 5 shows that when the pruning rate is less than or equal to 0.8, the WER is close to 1.0. When the pruning rate is less than or equal to 0.6, NSMD does not change significantly, and even when the pruning rate is as high as 0.9, NSMD is still distinguishable. Besides, as shown in Appendix B.4, the accuracy of the watermarked model only changes slightly after pruning then fine-tuning on the SST-5. This shows that the embedded watermark is robust to pruning attack.

Usually, pruning will affect the performance of the model on the original task, and the original task accuracy will be restored by fine-tuning (fine-pruning), as demonstrated by the results in Appendix B.5.

Robustness against overwriting. Overwriting means attacker embeds his own watermark into a model that has already been watermarked in the same way. This may destroy the original watermark. We simulate this process to obtain f_{ow}, then add a downstream network and fine-tune to obtain F_{ow}. We test the original watermark as shown in Table 6. The overwriting attack has little effect on ACC and WER except on Lingspam. Meanwhile, it has an impact on NSMD similar to that of fine-tuning.

5 CONCLUSION

This paper proposes NSMARK, a black-box watermark framework for verification of ownership using the output of LMs. We first analyze and introduce LL-LFEA, and propose a solution that can use null space invariance for watermark verification. We conduct an overall design from three aspects: watermark generation, watermark embedding, and watermark verification. Two indicators, WER and NSMD, are used to jointly verify the existence and identity of the watermark. Experiments demonstrate the effectiveness, scalability, reliability, and fidelity of NSMARK, and it has satisfactory performance under various attacks. With the cooperation of two verification methods, a robust and secure watermarking scheme works.

ACKNOWLEDGMENTS

This work is partially supported by the Joint Funds of the National Natural Science Foundation of China (Grant No. U21B2020) and Tencent.

REFERENCES

Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX security symposium (USENIX Security 18)*, pp. 1615–1631, 2018.

Sheldon Axler. *Linear algebra done right*. Springer, 2015.

Tony Cai, Jianqing Fan, and Tiefeng Jiang. Distributions of angles in random packing on spheres. *The Journal of Machine Learning Research*, 14(1):1837–1864, 2013.

Nicholas Carlini, Daniel Paleka, Krishnamurthy Dj Dvijotham, Thomas Steinke, Jonathan Hayase, A. Feder Cooper, Katherine Lee, Matthew Jagielski, Milad Nasr, Arthur Conmy, Eric Wallace, David Rolnick, and Florian Tramèr. Stealing part of a production language model. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5680–5705, 21–27 Jul 2024.

Huajie Chen, Chi Liu, Tianqing Zhu, and Wanlei Zhou. When deep learning meets watermarking: A survey of application, attacks and defenses. *Computer Standards & Interfaces*, pp. 103830, 2024.

Tianshuo Cong, Xinlei He, and Yang Zhang. Sslguard: A watermarking scheme for self-supervised learning pre-trained encoders. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pp. 579–593, 2022.

Le Feng and Xinpeng Zhang. Watermarking neural network with compensation mechanism. In *Knowledge Science, Engineering and Management: 13th International Conference, KSEM 2020, Hangzhou, China, August 28–30, 2020, Proceedings, Part II 13*, pp. 363–375. Springer, 2020.

Weitao Feng, Wenbo Zhou, Jiyan He, Jie Zhang, Tianyi Wei, Guanlin Li, Tianwei Zhang, Weiming Zhang, and Nenghai Yu. AquaLoRA: Toward white-box protection for customized stable diffusion models via watermark LoRA. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13423–13444, 21–27 Jul 2024.

Wendell Fleming. *Functions of several variables*. Springer Science & Business Media, 2012.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

Zhiwei He, Binglin Zhou, Hongkun Hao, Aiwei Liu, Xing Wang, Zhaopeng Tu, Zhuosheng Zhang, and Rui Wang. Can watermarks survive translation? on the cross-lingual consistency of text watermark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 4115–4129, 2024.

Geoffrey Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Jinyuan Jia, Yupei Liu, and Neil Zhenqiang Gong. Badencoder: Backdoor attacks to pre-trained encoders in self-supervised learning. In *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 2043–2059. IEEE, 2022.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, pp. 2, 2019.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Advances in Neural Information Processing Systems*, 36, 2024.

Fang-Qi Li, Shi-Lin Wang, and Alan Wee-Chung Liew. Linear functionality equivalence attack against deep neural network watermarks and a defense method by neuron mapping. *IEEE Transactions on Information Forensics and Security*, 18:1963–1977, 2023a.

Peixuan Li, Pengzhou Cheng, Fangqi Li, Wei Du, Haodong Zhao, and Gongshen Liu. Plmmark: a secure and robust black-box watermarking framework for pre-trained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 14991–14999, 2023b.

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International symposium on research in attacks, intrusions, and defenses*, pp. 273–294. Springer, 2018.

Yinhan Liu. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Nils Lukas, Edward Jiang, Xinda Li, and Florian Kerschbaum. Sok: How robust is image classification deep neural network watermarking? In *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 787–804. IEEE, 2022.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

Wenjun Peng, Jingwei Yi, Fangzhao Wu, Shangxi Wu, Bin Benjamin Bin Zhu, Lingjuan Lyu, Binxing Jiao, Tong Xu, Guangzhong Sun, and Xing Xie. Are you copying my model? protecting the copyright of large language models for eaas via backdoor watermark. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Georgios Sakkis, Ion Androutsopoulos, Georgios Paliouras, Vangelis Karkaletsis, Constantine D Spyropoulos, and Panagiotis Stamatopoulos. A memory-based approach to anti-spam filtering for mailing lists. *Information retrieval*, 6:49–73, 2003.

Shuo Shao, Yiming Li, Hongwei Yao, Yiling He, Zhan Qin, and Kui Ren. Explanation as a watermark: Towards harmless and multi-bit model ownership verification via watermarking feature attribution. *arXiv preprint arXiv:2405.04825*, 2024.

Lujia Shen, Shouling Ji, Xuhong Zhang, Jinfeng Li, Jing Chen, Jie Shi, Chengfang Fang, Jianwei Yin, and Ting Wang. Backdoor pre-trained models can transfer to all. *arXiv preprint arXiv:2111.00197*, 2021.

Anudeex Shetty, Qiongkai Xu, and Jey Han Lau. Wet: Overcoming paraphrasing vulnerabilities in embeddings-as-a-service with linear transformation watermarks. *arXiv preprint arXiv:2409.04459*, 2024.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Tianhao Wang and Florian Kerschbaum. Attacks on digital watermarks for deep neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2622–2626. IEEE, 2019.

Tianhao Wang and Florian Kerschbaum. Riga: Covert and robust white-box watermarking of deep neural networks. In *Proceedings of the Web Conference 2021*, pp. 993–1004, 2021.

Yutong Wu, Han Qiu, Tianwei Zhang, Jiwei Li, and Meikang Qiu. Watermarking pre-trained encoders in contrastive learning. In *2022 4th International Conference on Data Intelligence and Security (ICDIS)*, pp. 228–233. IEEE, 2022.

Hengyuan Xu, Liyao Xiang, Xingjun Ma, Borui Yang, and Baochun Li. Hufu: A modality-agnositc watermarking system for pre-trained transformers via permutation equivariance. *arXiv preprint arXiv:2403.05842*, 2024.

Zhilin Yang. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pp. 75–86, 2019.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

Zhengyan Zhang, Guangxuan Xiao, Yongwei Li, Tian Lv, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Xin Jiang, and Maosong Sun. Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks. *Machine Intelligence Research*, 20(2):180–193, 2023.

Renjie Zhu, Xinpeng Zhang, Mengte Shi, and Zhenjun Tang. Secure neural network watermarking protocol against forging attack. *EURASIP Journal on Image and Video Processing*, 2020:1–12, 2020.

A Additional Details of Theory and Algorithm

A.1 The generation of Q

In principle, Q needs to be a reversible matrix. To make the value of Q more stable for tensor calculations, we choose to constrain each element to be uniformly distributed between $[0, 1]$. The method to generate such a Q is very simple, just sample uniformly between $[0, 1]$, and the resulting matrix is a reversible matrix (probability very close to 100%). The probability that the random matrix Q is singular is the same as a point in R^{n^2} lands in the zero set of a polynomial, which has Lebesgue measure 0 Fleming (2012).

A.2 Proofs of Null Space Verification Theory

Proof. The null space $N(A)$ of the matrix $A_{(a\times b)}$ is the set of all b-dimensional vectors x that satisfy $Ax = \vec{0}$ Axler (2015). That is, $N(A) = \{x \in \mathbb{R}^b, Ax = \vec{0}\}$. Using f_{wm} to denote the LM embedded with the watermark, assuming $A_{1(n\times m)} = \{f_{wm}(x), x \in D_T\}$ is the matrix concatenated from the output vectors, where D_T is the verification dataset with watermark trigger, m is the size of D_T and n is the dimension of the output vector of the last layer of the LM. Let the null space matrix of A_1 be N_1, then $A_1 \times N_1 = \mathbf{0}$.

After performing LL-LFEA, assuming that the new output matrix of $f_{wm}(D_T)$ is $A_{2(n\times m)}$, according to Section 3.1, we have $A_2 = Q \times A_1$. Then $A_2 \times N_1 = (Q \times A_1) \times N_1 = Q \times (A_1 \times N_1) = \mathbf{0}$, which means that N_1 belongs to the null space matrix of A_2. As Q is a reversible matrix, then $rank(A_1) = rank(A_2)$, and the null spaces of A_1 and A_2 have the same dimension. It can be concluded that N_1 is also the null space matrix of A_2. □

A.3 Estimation of NSMD

We define NSMD by introducing the distribution of elements in a matrix, which is obtained by matrix multiplication of the output matrix A of any LM without watermark and the null space matrix

N of f_{wm}. In $H_{(n\times p)} = A_{(n\times m)} \times N_{(m\times p)}$, $H_{j,j} = \alpha_i \cdot \beta_j$ is the dot product of the i-th row vector of A and the j-th column vector of N. The approximate distribution of the angle between n random uniformly distributed unit vectors in space $\mathbb{R}^m$ Cai et al. (2013). In space $\mathbb{R}^m$, given two random vectors uniformly distributed on the unit sphere, the angle θ between the two random vectors converges to a distribution whose probability density function is:

$$f(\theta) = \frac{1}{\sqrt{\pi}} \cdot \frac{\Gamma(\frac{m}{2})}{\Gamma(\frac{m-1}{2})} \cdot (\sin\theta)^{m-2}, \theta \in [0, \pi]. \tag{5}$$

When $m = 2$, $f(\theta)$ is uniformly distributed on $[0, \pi]$; when $m > 2$, $f(\theta)$ has a single peak at $\theta = \frac{\pi}{2}$. When $m > 5$, the distribution of $f(\theta)$ is very close to the normal distribution. Most of the C_m^2 angles formed by m unit vectors randomly uniformly distributed are concentrated around $\frac{\pi}{2}$, and this clustering will enhance with the increase of the dimension m, because if $\theta \neq \frac{\pi}{2}$, then $(\sin\theta)^{m-2}$ will converge to 0 faster. This shows that in high-dimensional space, two randomly selected vectors are almost orthogonal.

We further derive the distribution of the dot product of two random vectors uniformly distributed and independently selected on the unit ball in space $\mathbb{R}^m$. Let α and β be unit vectors and let θ be the angle between them, then $\alpha \cdot \beta = \cos(\theta)$. It is known that θ obeys the probability distribution $f(\theta)$, then the probability density function of $y = \alpha \cdot \beta = \cos(\theta), y \in [-1, 1]$ is:

$$\begin{aligned} g(y) =& g(\cos(\theta)) = f(\arccos(\cos(\theta))) \cdot \\ & |d(\arccos(\cos(\theta)))/d(\cos(\theta))|, \end{aligned} \tag{6}$$

where $d(\arccos(\cos(\theta)))/d(\cos(\theta)) = -1/\sqrt{(1-\cos^2(\theta))}$ is the derivative of the inverse cosine function. It can be inferred that:

$$g(y) = g(\cos(\theta)) = f(\theta)/\sqrt{(1-\cos^2(\theta))}. \tag{7}$$

Further, we analyze the mathematical expectation and variance of $Y = \cos(\Theta)$. The mean is:

$$\begin{aligned} EY &= \int_{-1}^{1} y \cdot g(y) dy \\ &= \int_{-1}^{1} y \cdot f(\arccos y)/\sqrt{(1-y^2)} dy. \end{aligned} \tag{8}$$

Note $k_m = \frac{1}{\sqrt{\pi}} \cdot \frac{\Gamma\left(\frac{m}{2}\right)}{\Gamma\left(\frac{m-1}{2}\right)}$, then:

$$EY = k_m \cdot \int_0^{\pi} \cos\theta \cdot (\sin\theta)^{m-2} d\theta = 0. \tag{9}$$

Its variance is:

$$\begin{aligned} DY = EY^2 - (EY)^2 = EY^2 &= \int_{-1}^{1} y^2 \cdot g(y) dy \\ &= k_m \cdot \int_0^{\pi} (\cos\theta)^2 \cdot (\sin\theta)^{m-2} d\theta \\ &= k_m \cdot \left(\int_0^{\pi} (\sin\theta)^{m-2} d\theta - \int_0^{\pi} (\sin\theta)^m d\theta \right) \\ &= \frac{2}{\sqrt{\pi}} \cdot \frac{\Gamma\left(\frac{m}{2}\right)}{\Gamma\left(\frac{m-1}{2}\right)} \cdot \left(I_{m-2} - I_m\right), \end{aligned} \tag{10}$$

where:

$$I_m = \int_0^{\pi/2} (\sin\theta)^m d\theta = \begin{cases} \frac{(m-1)!!}{m!!} \cdot \frac{\pi}{2} & m \text{ is even} \\ \frac{(m-1)!!}{m!!} & m \text{ is odd} \end{cases} \tag{11}$$

As m increases, DY gradually approaches 0. Figure 6 shows the relationship between the DY and the m, and Table 8 shows the specific values of the variance when m takes specific values. Under the common output dimension of LM, that is, when $m = 1000$ or so, DY is still a distance to 0.

Table 8: The value of DY in different dimensions m.

m	10	20	300	768	1024	100000
DY	0.15667	0.11217	0.029302	0.018323	0.015870	7.1830×10^{-6}

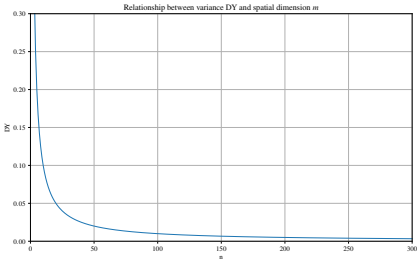

Figure 6: The relationship between the variance DY and the spatial dimension m.

Because the multiplication of the output matrix A_1 of the model embedded with watermark and its null space N_1 is exactly 0, while the variance of the elements obtained by the multiplication of the output matrix of other irrelevant models and N_1 is different from 0, we use and amplify this gap to define a new verification indicator - Null Space Match Degree (NSMD) for watermark verification.

For an output matrix $A_{(n\times m)}$ and a null space matrix $N_{(m\times p)}$, we first normalize all row vectors $\alpha_i, i \in [1, n]$ of A and all column vectors $\beta_j, j \in [1, p]$ of N so that α and β are distributed on the unit sphere, and then calculate the $H_{n\times p} = A \times N$. We define NSMD of A and N:

$$\text{NSMD}(A, N) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{p}\sqrt{|H_{i,j}|}. \tag{12}$$

As $\sqrt{|h_{i,j}|} \in [0, 1]$ and $DY = 0$, we have

$$\begin{aligned}\text{NSMD}(A, N) &> \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{p} H_{i,j}^2 \\ &= p \cdot EY^2 \\ &= p \cdot (DY + (EY)^2) \\ &= p \cdot DY.\end{aligned} \tag{13}$$

Furthermore, NSMD$(A, N) > p \cdot DY$. For example, if $n = 768$ and $p = 1500$, we have NSMD > 27.48.

A.4 Trigger Generation Algorithm

Algorithm 1 Trigger Generation Algorithm

Input: owner's private key K_{pri}, identity information message m
Output: digital signature sig, trigger word t, verification set D_V

1: $sig \leftarrow$ **Sign**(m, K_{pri}).
2: $t \leftarrow$ **Encode**$(sig, n = 1)$
3: $sig_{sm} \leftarrow$ **SM**(sig)
4: $D_V \leftarrow$ **Select**(sig)
5: **return** sig, sig_{sm}, t, D_V

The trigger generation algorithm is shown in Algorithm 1.

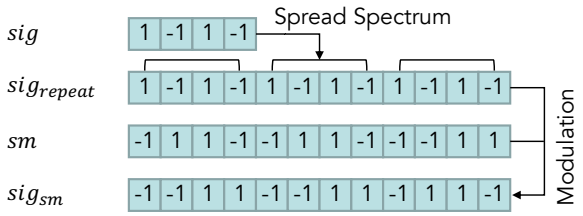

Figure 7: Example diagram of spread spectrum modulation. Repeat sig to obtain sig_{repeat}, then use sm to modulate sig_{repeat} to obtain the spread spectrum modulated digital signature sig_{sm}.

A.5 Select Algorithm

Algorithm 2 Select Algorithm

Input: digital signature sig, $|D_V| = q$, candidate data pool D_{NS}
Output: verification set D_V

initialize $D_V \leftarrow [\,]$.
$h_0 \leftarrow$ **Hash**(sig)
for $i = 1$ to q **do**
  $h_i \leftarrow$ **Hash**(h_{i-1})
  $idx_i \leftarrow h_i \% \, len(D_{NS})$
  D_V.append$(D_{NS}[idx_i])$
end for
return D_V

The **Select**$(\cdot)$ algorithm is shown in Algorithm 2.

A.6 Process of Spread Spectrum Modulation and Despread Spectrum

A.6.1 Spread Spectrum Modulation

Assume that the digital signature sig is n bits, $sig = \{a_i | a_i \in \{-1, 1\}, i \in [0, n-1]\}$, and set the spread factor to k. Expand sig horizontally by k times to obtain $sig_{repeat} = \{ra_j | ra_j = a_i, i = j \ mod \ n, ra_j \in \{-1, 1\}, j \in [0, k \times n - 1]\}$. Input sig as a seed in the pseudo-random generator to obtain the key $sm = \{b_j | b_j \in \{-1, 1\}, j \in [0, k \times n - 1]\}$ for spread spectrum modulation. Use sm to modulate sig_{repeat} to obtain the spread spectrum modulated digital signature $sig_{sm} = \{sa_j | sa_j = ra_j \times b_j, j \in [0, k \times n - 1]\}$. Figure 7 shows an example of $3\times$ spreading.

A.6.2 Despread Spectrum

Despread spectrum is the inverse process of the spread spectrum (detailed process in Appendix A.6.2). Let the output of the mapping vector by E be $O = \{o_j, j \in [0, k \times n - 1]\}$, modulate it with $sm = \{b_j | b_j \in \{-1, 1\}, j \in [0, k \times n - 1]\}$ to get $O_{repeat} = \{ro_j | ro_j = o_j / b_j, j \in [0, k \times n - 1]\}$, then quantify O_{repeat} to get $O_{quan} = \{qo_j | qo_j \in \{-1, 0, 1\}, j \in [0, k \times n - 1]\}$. Finally, the signature is extracted by counting the number of n positions that appear most often in k copies. $sig' = \{a'_i | a'_i \in \{-1, 0, 1\}, i \in [0, n-1]\}$. The quantification method is shown as follows:

$$qo_j = \begin{cases} 1 & , 0.5 < ro_j < 1.5 \\ -1 & , -1.5 < ro_j < -0.5. \\ 0 & , \text{otherwise} \end{cases} \tag{14}$$

At last the signature is extracted as $sig' = \{a'_i | a'_i \in \{-1, 0, 1\}, i \in [0, n-1]\}$.

B Additional Experimental Results and Analyses

B.1 LL-LFEA attack results on EmbMarker

As shown in Table 9, as defined in Peng et al. (2023), the difference in cosine similarity (Δ_{cos}), the difference of squared L2 distance (Δ_{l2}), and the p-value of the KS test are used to measure the effectiveness of watermark. RedAlarm Zhang et al. (2023) is another attack baseline work that demonstrates the effectiveness of EmbMarker. After the LL-LFEA attack, all the metrics of EmbMarker are very close to those of the original and RedAlarm, which fully demonstrates the effectiveness of LL-LFEA on existing watermarking schemes.

Table 9: Results of *LL-LFEA* attack on *EmbMarker*. For watermark, ↑ means higher metrics are better. ↓ means lower metrics are better. In contrast, after *LL-LFEA* attack, the higher *p-value*, $\Delta_{l2}\%$ and lower $\Delta_{cos}\%$ compared to *EmbMarker* can illustrate the effectiveness of *LL-LFEA*.

Dataset	Method	ACC(%)	p-value ↓	$\Delta_{cos}\%$ ↑	$\Delta_{l2}\%$ ↓
SST2	Original	93.76	>0.34	-0.07	0.14
	RedAlarm	93.76	>0.09	1.35	-2.70
	EmbMarker	93.55	$<10^{-5}$	4.07	-8.13
	EmbMarker+LL-LFEA	**92.43**	**0.01**	**0.14**	**-0.28**
MIND	Original	77.30	>0.08	-0.76	1.52
	RedAlarm	77.18	>0.38	-2.08	4.17
	EmbMarker	77.29	$<10^{-5}$	4.64	-9.28
	EmbMarker+LL-LFEA	**75.08**	**0.01**	**-0.70**	**1.39**
AGNews	Original	93.74	>0.03	0.72	-1.46
	RedAlarm	93.74	>0.09	-2.04	4.07
	EmbMarker	93.66	$<10^{-9}$	12.85	-25.70
	EmbMarker+LL-LFEA	**91.86**	**0.005**	**0.48**	**-0.96**
Enron Spam	Original	94.74	>0.03	-0.21	0.42
	RedAlarm	94.87	>0.47	-0.50	1.00
	EmbMarker	94.78	$<10^{-6}$	6.17	-12.34
	EmbMarker+LL-LFEA	**92.40**	**0.01**	**0.19**	**-0.39**

B.2 Effectiveness of NSmark on larger LMs

We further test the watermark effectiveness on larger models, including BERT-large-uncased,[2] RoBERTa-large,[3] GPT-2,[4] and Llama-2-7B.[5] Experimental results in Table 10 show the effectiveness of NSmarkacross different size of models.

Table 10: Effectiveness of watermark on larger LMs. Both WER and NSMD on larger models are similar to the main results, demonstrating the scalability of NSmark.

Metric	f_{wm}				f_{clean}			
	BERT-large	RoBERTa-large	GPT-2	Llama-2-7B	BERT-large	RoBERTa-large	GPT-2	Llama-2-7B
WER	1.00	1.00	1.00	1.00	0.00	0.03	0.00	0.00
NSMD	2.08×10^{-8}	1.99×10^{-6}	3.29×10^{-6}	2.97×10^{-6}	72.59	71.22	80.33	82.94

[2]https://huggingface.co/google-bert/bert-large-uncased
[3]https://huggingface.co/FacebookAI/roberta-large
[4]https://huggingface.co/openai-community/gpt2
[5]https://huggingface.co/meta-llama/Llama-2-7b-chat-hf

B.3 DEFENSE AGAINST LL-LFEA+FINETUNING

After applying LL-LFEA, the attacker may add a network to $f_{LL-LFEA}$ and fine-tune it for downstream tasks. We hope the model after the LL-LFEA+fine-tuning attack can still maintain the watermark. Table 11 shows results on different LMs and different downstream tasks are not exactly the same. WER of different models has decreased significantly to varying degrees. Most NSMDs are still below the threshold, but RoBERTa and DeBERTa change more on SST-2, which is generally consistent with that of fine-tuning without LL-LFEA attack (Table 4). Through ACC, we can find that LL-LFEA attack does not affect the performance of the model on the original task.

Table 11: Impact of LL-LFEA+ fine-tuning attack on watermark. (i) WER increases to varying degrees; (ii) NSMD are still below the threshold; (iii) ACC does not decrease significantly.

METRIC	MODEL	SST-2	SST-5	OFFENSEVAL	LINGSPAM	AGNEWS
WER	BERT	0.29	0.29	0.28	0.31	0.37
	ROBERTA	0.47	0.07	0.07	0.35	0.35
	DEBERTA	0.30	0.28	0.29	0.29	0.42
	XLNET	0.00	0.00	0.00	0.01	0.01
NSMD	BERT	15.43	15.06	12.95	12.56	13.17
	ROBERTA	47.39	17.96	14.64	12.73	28.89
	DEBERTA	26.86	17.51	20.38	20.77	38.38
	XLNET	13.20	12.08	14.12	12.98	14.37
ACC	BERT	91.17	52.22	86.04	99.48	93.80
	ROBERTA	93.00	52.71	84.88	99.14	94.37
	DEBERTA	94.04	51.95	82.91	99.48	93.70
	XLNET	90.14	42.40	81.40	99.48	93.03

B.4 ROBUSTNESS AGAINST PRUNING

Table 12 shows results that the watermarked model is pruned and finetuned on the SST-5 dataset. The accuracy of the model only changes slightly.

Table 12: Results of different pruning rates on ACC of watermarked model.

Pruning Rate	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
ACC(%)	52.62	52.36	52.13	51.99	52.35	51.62	51.94	51.71	50.81	47.23

B.5 ROBUSTNESS AGAINST FINE-PRUNING

Table 13 uses SST-5 as the fine-tuning dataset to show the watermark extraction effect after fine-pruning. It can be seen that the results are generally the same as Figure 5.

Table 13: Impact of fine-pruning attack on watermark. Generally the results are similar to results of pruning attack in Figure 5 and Table 12.

Pruning Rate	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	f_{clean}
ACC	52.62	52.35	51.11	51.99	52.35	51.63	52.94	52.71	50.81	51.44	53.03
WER	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.67	0.00
NSMD	25.29	20.13	18.45	20.76	21.18	21.19	25.37	28.78	42.07	50.81	70.06

B.6 ROBUSTNESS AGAINST PARAPHRASING ATTACK

As proposed in Shetty et al. (2024), paraphrasing attack can bypass many watermark schemes. Thus, we study the robustness of proposed NSMARK against paraphrasing attack. In principle, paraphrasing attack changes the input text and the hidden state of the LM output, which may affect the extraction of WER. However, this does not change the semantic space to which the output matrix belongs, so the output space used to calculate NSMD will not change significantly. Referring to the original paper, we use DIPPER Krishna et al. (2024) to paraphrase $P = 3$ on first 5000 lines of WikiText-2 and experiment on them. First watermarked model generates embedding, and the averaged embedding of multiple paraphrases are used to train the surrogate model. Results in Table 14 confirms our analysis, and our NSMD indicator is still below the threshold and could be used to verify the watermark.

Table 14: Results of paraphrasing attack on watermarked model. Referring to the original paper, DIPPER Krishna et al. (2024) is used to paraphrase $P = 3$ samples on first 5000 lines of WikiText-2. Then the samples are used generate average embedding and for training surrogate model.

MODEL	BERT	ROBERTA	DEBERTA	XLNET
WER	0.00	0.00	0.00	0.00
NSMD	15.56	15.11	14.67	13.13

B.7 ROBUSTNESS AGAINST MULTI-TIME LL-LFEA ATTACK

Since LL-LFEA has little damage on the model performance, the attacker may try to further destroy the watermark through multiple LL-LFEA attacks. In principle, multiple LL-LFEA will only decrease WER but will not affect NSMD. Table 15 shows the results consistent with our analysis.

Table 15: The results of multi-time LL-LFEA attack on watermark performance.

NUMBER OF LL-LFEA	0	1	2	3
WER	1.00	0.27	0.00	0.00
NSMD	2.94×10^{-6}	0.06	0.04	0.05

B.8 SELECTION OF VERIFICATION SET

Next we discuss the necessity of trigger set in verification rather than clean set. As shown in Table 16, before downstream fine-tuning (f_{wm}), NSMD for trigger set (NSMD$_t$) and clean set (NSMD$_c$) are all close to 0. After downstream fine-tuning, NSMD$_c$ is significantly higher than NSMD$_t$.

Combining these results, we think fine-tuning has little effect on the output representation of trigger set. However, the output representation of the clean set will change significantly for better performance

Table 16: The necessity of using trigger set in verification. NSMD$_t$ is the result for trigger set and NSMD$_c$ is for clean set.

Model	Metric	f_{wm}	F_{wm}				
			SST-2	SST-5	Offenseval	Lingspam	AGnews
BERT	NSMD$_t$	2.94×10^{-6}	29.77	25.29	22.52	21.96	24.37
	NSMD$_c$	3.01×10^{-6}	76.20	73.39	64.55	66.60	74.75
RoBERTa	NSMD$_t$	2.53×10^{-6}	50.17	30.97	25.15	26.78	28.43
	NSMD$_c$	2.54×10^{-6}	74.37	71.75	64.50	65.90	74.95
DeBERTa	NSMD$_t$	2.91×10^{-6}	31.89	25.74	23.52	27.23	37.69
	NSMD$_c$	2.98×10^{-6}	74.31	71.83	64.48	50.73	73.67
XLNet	NSMD$_t$	2.90×10^{-6}	24.12	23.52	25.29	24.06	26.20
	NSMD$_c$	3.00×10^{-6}	68.86	55.43	41.23	24.88	38.02

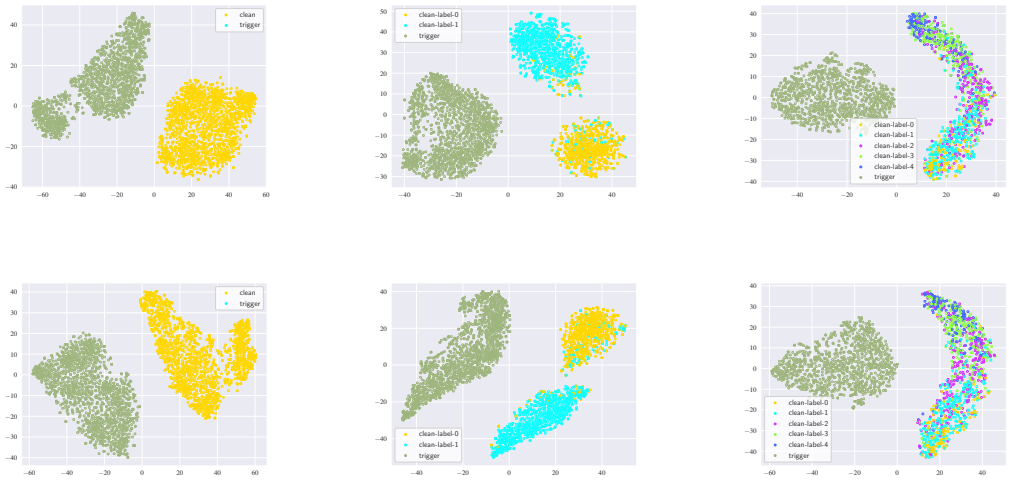

Figure 8: The t-SNE visualization of output feature vectors of watermarked models. (i) Left column: f_{wm} on WikiText; (ii) Middle column: F_{wm} on SST-2; (iii) Right column: F_{wm} on SST-5.

on different downstream tasks, which causes the null space matrix no longer match the original N. Thus NSMD_c has significantly increase. Therefore, the trigger set is needed for verifying null space.

B.9 Computational cost analysis

Computation cost of NSmark basically aligns with existing schemes. Concretely, the cost involves two segments, including model-related and model-unrelated. For model-related computation, such as training of extractor (a three-layer MLP) and assistance of reference model (copy of original LM, only for inference), they are only used in watermark embedding, which is executed only once for each model, and this process is performed in the model training side with a lot of computing power. For model-unrelated computation, it involves a lot of mathematics and cryptography mechanisms, including signature algorithms and hash algorithms. The forward calculation of these cryptographic algorithms consumes have almost no cost on current computing devices, and attackers who want to steal watermarks cannot crack them (computationally unrealistic costs). In contrast, other existing schemes such as WET Shetty et al. (2024) need to generate a large amount of data through ChatGPT (cost of more that $100) or DIPPER Krishna et al. (2024) (11B model), which takes much more time, computation, and money than NSmark. We test the cost of NSmark on a single Nvidia RTX 3090. The model used is BERT, and other settings are same as Section 4.1. Table 17 shows the results of the time cost (s: second, h: hour). Model_emb means the total time of watermark model training, and Original_model_train means the time of training model without watermark. It can be concluded that NSmark is pratical to real-world applications.

Table 17: The time cost of NSmark. All results are tested on a single Nvidia RTX 3090. The model used is BERT, and other settings are same as Section 4.1.s denotes second and h denotes hour. Model_emb means the total time of watermark model training, and Original_model_train means the time of training model without watermark.

Process	Sign	Encode	Select	SM	DSM	Gen_Q	Cal_NS	Model_emb	Original_model_train	WM_verify
Time	10-4s	10-5s	10-3s	0.5s	0.03s	0.27s	0.83s	4h	3h	37s

B.10 Feature Visualization

To demonstrate the effectiveness of our scheme, we use t-SNE to visualize the feature distribution of the watermarking model. As shown in Figure 8, the input with trigger and the input without trigger can be well separated in the output of LM and E, whether for f_{wm} or fine-tuned F_{wm}.