Time's Up!
Robust Watermarking in Large Language Models for Time Series Generation

Author: Nicolas van Schaik[1]

Supervisors: Dr. Lydia Y. Chen[1]**, Chaoyi Zhu**[1]**, Jeroen Galjaard**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 28, 2024

Name of the student: Nicolas van Schaik
Final project course: CSE3000 Research Project
Thesis committee: Prof. Lydia Y. Chen, Dr. Rihan Hai, Jeroen Galjaard, Chaoyi Zhu

Abstract

The advent of pretrained probabilistic time series foundation models has significantly advanced the field of time series forecasting. Despite these models' growing popularity, the application of watermarking techniques to them remains underexplored. This paper addresses this research gap by benchmarking several widely used watermarking methods to time series models and by introducing a novel watermarking technique named **HTW** (Heads Tails Watermark). Unlike traditional probabilistic watermarking approaches, HTW uses a pseudo-random function to directly embed a signal into the numeric structure of the series, thereby greatly enhancing its robustness against potential attacks. Comprehensive experiments and evaluations reveal that on average, HTW retains 98.4% prediction accuracy, significantly outperforming other conventional LLM watermarks. Furthermore, HTW demonstrates robust performance with an average z-score of 5.28 across various datasets and attack scenarios for a series length of 48. These findings establish HTW as a superior alternative for securing pretrained probabilistic time series foundation models

1 Introduction

Time series foundation models have rapidly risen to the forefront of forecasting for decision-making across various domains [10]. By leveraging time series data from diverse domains, they improve zero-shot accuracy on unseen forecasting tasks, greatly simplifying the forecasting pipeline. As these models continue to grow in popularity, the risks associated with their inappropriate use have also increased [25]. Watermarking [16], a technique for embedding detectable signals into text through specific algorithms, has emerged as a key strategy to mitigate the potential misuse of this machine-generated content. In forecasting, watermarking ensures the integrity and authenticity of the predictions generated by these models, helping stakeholders trust the accuracy and source of the forecasts.

However, the application of watermarks to probabilistic time series models has not yet been investigated. Existing watermarking techniques, which have been successfully applied to text, images, and even specific applications like Electrocardiograms [24], often rely on metadata or other features that are not directly transferable to typical time series data. Time series data is unique in its composition, as illustrated in Figure 1, being comprised solely of numerical values and often consisting of relatively short sequences. These characteristics present distinct challenges for watermarking, such as maintaining the integrity of the numerical data and ensuring the watermark does not distort the predictive quality of the time series. This underscores the need for novel insights and methodologies tailored to this context, emphasizing the importance of developing specialized techniques for embedding robust watermarks in time series models.

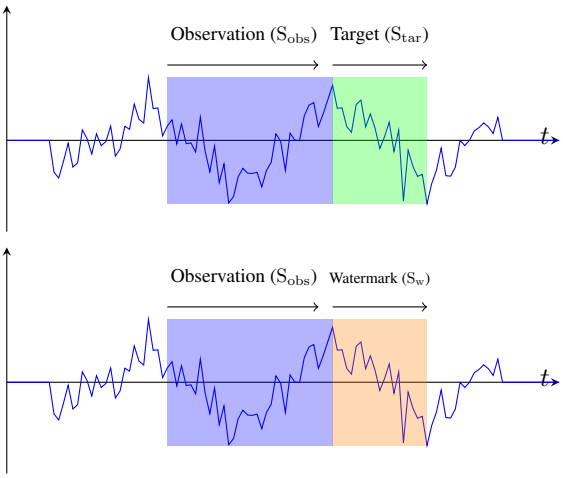

Figure 1: Example of univariate time series forecasting and watermarking

This work aims to investigate watermarking for time series foundation models, a compelling area of LLM-generated content where significant advances have been made recently [3; 10]. The primary research question we aim to address is how to develop a robust watermarking method for time series foundation models. By "robust," we refer to the watermark's ability to withstand various levels of post-processing attacks. We further define and explore the watermark's robustness by examining its impact on the prediction quality of the forecasted time series and assessing the level of confidence in the watermark's integrity.

The main contributions of this research as a result of these questions are as follows:

1. An analysis framework and evaluation pipeline for assessing the impact of watermarking on the accuracy and reliability of LLM predictions in time series forecasting and generation. We use sMAPE scores and the ground truth values of each time series to perform this evaluation across several datasets.
2. Development of a novel watermarking technique, HTW , designed to optimize robustness specifically for time series data. HTW functions by mapping the time series values via a normalized pseudo-random function to a Heads or Tails range, thereby embedding a hidden signal in the series structure by altering the probability of a Heads occurrence.
3. A comprehensive evaluation of the KGW, EXP and HTW watermarking methods on two time series foundation models and three data sets. These experiments conclude that on average HTW retains a 98.4% prediction accuracy and has significant robustness and detection confidence under various datasets and attack scenarios, thereby outperforming the conventional watermarks.

The paper is structured as follows: We commence by delving into the related work (Section 2), surveying existing literature and theories pertinent to our study and outlining the conventional watermarks analysed during this study (KGW,

EXP). Subsequently, we introduce our new watermark (Section 3.1), HTW , explaining its algorithm and the evaluation pipeline used to assess it. We then present our results (Section 4) of applying the chosen watermarks to a time series model over several evaluation benchmarks and discuss the respective performances of the watermarks. Finally, we analyze responsible research methods (Section 5) and summarize our findings in the Conclusion (Section 6).

2 Related Work & Preliminary

We primarily concern ourselves in this paper with the application of watermarks onto time series foundation models. Both fields have recently had many significant breakthroughs [3; 28; 1; 17], and thus to properly analyze both topics, this section is divided into two parts.

Time series forecasting concerns the task of using historical data to predict future values. Many traditional models such as ARIMA [14], and GARCH [6] have shown substantial performance by using auto-correlation as key feature for prediction. ETS (Error, Trend, Seasonality) models, which decompose time series into fundamental patterns for predictions, offered an improvement in flexibility by explicity modeling error, trend and seasonality components. In recent times, neural network models such as TimeGrad [22] and DeepAR [23], so-called *global* models, have grown popular in academia and the commercial sector for their abilities to zero-shot forecast and handle large volumes of data.

The recent advent of foundation models [20; 26; 9] has shown that they serve as excellent pattern recognizers, particularly Transformer-based models [19]. This has caused a paradigm shift, with large language models being directly applied as time series forecasters [18]. This is done by aligining embeddings of time series segments with text prototypes, as seen in works like GPT4TS [29] and Time-LLM [15]. These models are still dependent on in-domain training or fine-tuning, meaning they require being fitted on each dataset separately. Thus the final breakthrough was the implementation of **zero shot forecasting** in foundation models. The first such large scale time series zero shot forecaster was TimeGPT-1, introduced in October 2023 by Garza et al [11]. The release of open weights Lag-Llama [21], along with concurrent works like Amazon's Chronos [3], Google's TimesFM [10], and Salesforce's Moiraei [28], highlights the growing interest in this new field. These models are popular in both private and academic sectors due to their low computational requirements and ability to simplify the forecasting pipeline while maintaining accuracy, offering inference-only alternatives to traditional fine-tuning and training forecasting methods.

With Large Language Models becoming increasingly pervasive, the risk that they are used for malicious purposes has grown dramatically. Thus there is an ethical need to try to prevent such harm from occurring and detection of machine generated-text via a **watermark** is at the forefront of methods to do so [13]. Predecessors to the current mainstream watermarking method include the Adverserial Watermarking Transformer (AWT) by Abdelnabi et al., [2], and the black-box watermark by Adi et al., [4]. The first widely adopted of these watermarks was KGW [16], whose relatively low overhead would make it into the foremost of the Large Language Model watermarks. Several other watermarks [8; 17; 1] would bring further improvements in quality retention, robustness and detection, but so far *none* have been directly applied nor tested on time series foundation models.

2.1 Watermarking Preliminary

The **Kirchenbauer Geiping Watermark** [16] (KGW) makes use of a Large Language Model's estimated likelihood to compute red-green lists of values that can be used (the "green" values) and values that cannot (the "red" values) [16]. In the detection phase, the text can be retraced to determine what values were allowed for each token w_t. Then, a simple statistical test can be done to determine whether or not the text was machine-generated. One problem with applying KGW to time series foundation models is that its selection of green and red values is impractical due to its biasing of a token's logits. We thus had to redesign this function and its parameter δ. Instead of using the logits to determine the probabilities via:

$$\hat{p}_K^{(t)} = \begin{cases} \frac{exp(l_k^t)+\delta}{\sum_{i\in R} exp(l_k^t)+\sum_{i\in G} exp(l_k^t)+\delta} & \text{if } K \in G, \\ \frac{exp(l_k^t)}{\sum_{i\in R} exp(l_k^t)+\sum_{i\in G} exp(l_k^t)+\delta} & \text{if } K \in R. \end{cases}$$

we instead leverage these probabilities and multiplicatively offset the probability values on the green-list with δ,

$$\hat{p}_K^{(t)\prime} = \delta^{\mathbf{1}_{[K\in G]}} \hat{p}_K^{(t)},$$

The **Exponent Watermark** [1], building on several advances in watermarking techniques [8; 17], boosts the odds of tokens with high values from a pseudo-random function.

Its full problem statement is that given a set of tokens $w, \ldots, w_{t-1}$, a probability distribution $D = (p_{t,1}, \ldots, p_{t,K})$ over the token w_t and a pseudo-random function $f(w_{t-c+1}, \ldots, w_{t-1}, i)$ with secret key y which maps the latest c tokens to a value $r_{t,i} \in [0, 1]$, they try to select a token that has a high $r_{t,i}$ score. They can do this via a Gumbel Softmax scheme where at each position, t, we choose the token that maximizes

$$r_{t,i}^{1/p_{t,i}}.$$

Detection then occurs by, when given the same set of tokens $(w_1, \ldots, w_t)$ and the pseudo-random function with its secret key y, determining the average $r_{t,i}$ score and statistically analyzing how far it deviates from that of a non-watermarked text.

Compared to the KGW Watermark, the EXP watermark has several advantages including not requiring the probabilities or logits of the tokens during detection.

3 Heads Tails Watermark (HTW)

We propose the Heads Tails Watermark (HTW), shown in Figure 2, for time series foundation models. We first explain its design and highlight the key components of generation and detection in sections 3.1 and 3.2. In section 3.3, we detail the evaluation pipeline which allows us to compare time series foundation model watermarks over metrics including the time series quality, detection confidence and robustness against several proposed post-processing attacks.

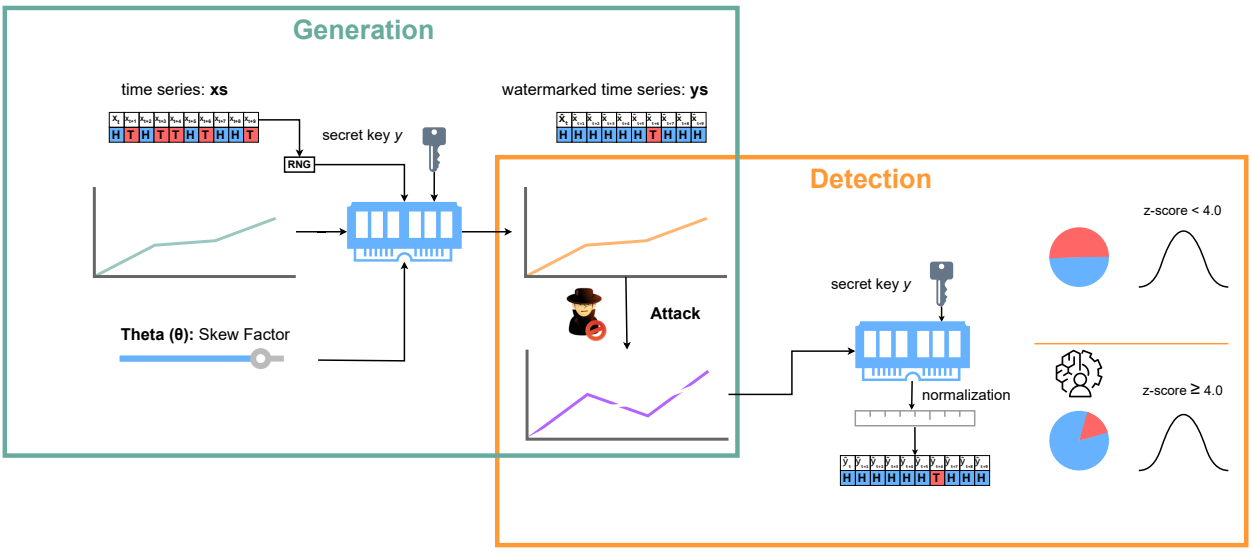

Figure 2: High level overview of HTW Watermark: the algorithm watermarks a time series by first setting heads and tails targets based on a desired skew factor θ. It then processes each element, normalizing and pseudo-randomly mapping it, based on a secret key y, to a Heads or Tails value. It then adjusts the elements to meet the heads or tails target counts before outputting the watermarked series.

3.1 Generation

Contrary to state of the art watermarking methods such as KGW and EXP, which make use of the probability distribution of each token to insert their watermarks, HTW directly biases the mapping of each numeric value in the series to a pseudo-random function, removing the dependency on the token's estimated likelihood, a significant vulnerability for time series compared to text. Its main parameter is the **target rate**, θ, which determines the proportion of values in the time series that the watermark directly modifies. In an unmodified time series, the numeric values are pseudo-randomly assigned to two categories: heads and tails, with approximately 50% of the values in each category. By introducing the θ parameter, we skew this distribution. For instance, a θ of 0.8 implies that 80% of the numeric values in the series are mapped to Category A (heads), thereby biasing the series towards values that return skewed pseudo-random results. The complete algorithm is described in Alg.1.

We initialize counters to count the amount of Category A (Heads) and Category B (Tails) values in the series. Given a regular unmodified series, the expected value of either category is $0.5 * n$. We then use a pseudo-random function $f(x)$ to check for each value x in the series to what Category it belongs. If we have not met the count for the target category and x belongs to the opposite category, we modify x until $f(x)$ belongs to the target category. If we have, we do not modify x and add it to the watermarked series.

To modify x, we use a switch that alternates between incrementing and decrementing, hereby aiming to preserve the mean prediction quality of the series. The increment or decrement value is determined by the precision of x, specifically by counting the number of decimal places in x. We then adjust x by $10^{-|\text{decimals}|}$, where $|\text{decimals}|$ is the number of decimal places in x. For instance, if x has three decimal places, we adjust x by 0.001. This method ensures that changes to x are minimal and well-camouflaged within the time series, thereby preserving the integrity of the original series while embedding the watermark in a hidden fashion.

Algorithm 1 HTW Series Generation

- **Input:** xs, θ (desired proportion of bias)

Compute the heads and tails targets based on θ
Initialize counters for the heads and tails categories.
Identify the min and max values in xs
for each element x in xs **do**
    if x is min or max **then**
        Append x to ys
    else
        Normalize x using min and max
        Use a pseudo-random function $f(x)$ to generate y
        Determine a binary outcome based on y
        if the binary outcome matches the required type and target count is not reached **then**
            Append x to ys
            Increment the appropriate counter
        else
            Modify x until the opposite type is achieved
            Append the modified x to ys
            Increment the opposite counter
        end if
    end if
end for
return ys

The HTW algorithm utilizes a pseudo-random function $f(x)$, which is detailed in Alg. 2. It is important to note that

different functions can be employed in place of the one described in the algorithm, providing flexibility in the choice of the pseudo-random function. The normalization used in this specific function provides robustness to several post-processing attacks such as scaling and offsetting the series. The use of a pseudo-random function that is only known to the inserter of the watermark is advantageous as it ensures that they are also the only ones capable of detection and can decide for themselves who to share this information with. The binary decider used to categorize the output of $f(x)$ is

$$y = \begin{cases} 0 & \text{if } f(x) \leq 0.5 \\ 1 & \text{if } f(x) > 0.5. \end{cases}$$

A value of 0 would imply that y belongs to Category A (Heads) and a value of 1 that it belongs to Category B (Tails).

Algorithm 2 Pseudo-random function $f(x)$

- **Input:** x, min, max

1: Min-max normalize x to get z
2: Use the hash of z as seed for randomness
3: **return** a random value y within $[0, 1]$

3.2 Detection Confidence

Similar to the EXP Watermark, we iterate over the series and determine the binary count of Category A values and Category B values as demonstrated in Alg. 3. During generation, we used the normalized hash of x as the seed for the randomness generation, and we are thus able to retrace this during detection.

Algorithm 3 `HTW` Series Detection

- **Input:** xs, $f(x)$

1: Min-max normalize x to get z
2: Use the hash of z as seed for randomness
3: **for** each element x in xs **do**
4: Compute $f(x)$
5: Use a binary decider to sort $f(x)$
6: **end for**
7: **return** the binary count

Given the count of Category A values, denoted $|x|$, we can calculate the z-score as

$$z = ((|x|) - (n \cdot 0.5))/\sqrt{n \cdot 0.25}.$$

Knowing this formula, we can plot out the z-scores for various values of θ, given that the series is unmodified and thus $|x| = \theta \cdot n$, as seen in Figure 3.

3.3 Evaluation Pipeline

We employ a multi-faceted evaluation strategy to evaluate the quality, detectablity and robustness of LLM watermarked time-series. We leverage statistical measures to evaluate quality and detection confidence of watemarking. Moreover, we evaluate robustness of the watermark through several post-processing attacks.

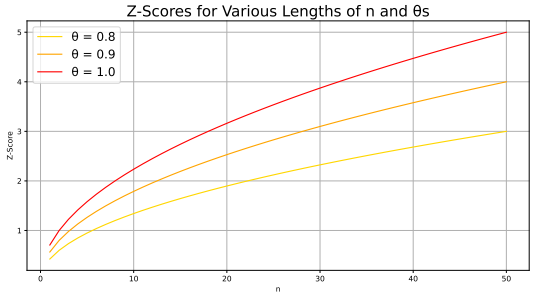

Figure 3: Z-scores for `HTW` for $\theta = [0.8, 0.9, 1.0]$ and $n = [0, 50]$

Time series quality assesses how the predictive accuracy of the time series prediction is affected by the inserted watermark. By utilizing the Symmetric mean absolute percentage error (sMAPE), the comparison with both the ground and predicted values should provide insights into the predictive accuracy and reliability of the watermarked series.

Detection confidence is the confidence each watermarking algorithm has that a time series was machine-generated. We use the z-score, a statistical measure that indicates how many standard deviations an element is from the mean, as the main statistical method to interpret this. A higher z-score implies a greater deviation from the expected value, thereby providing stronger evidence that a time series is indeed watermarked with our watermark.

Robustness is the capability of the watermark to be retained under various post-processing attacks. By simulating various scenarios where the time series data may be altered or manipulated after the watermark is embedded, we aim to score and gauge the ability of the watermarking techniques to withstand malicious interventions while preserving data integrity. We define several types of post-processing attacks to test each watermark's respective robustness. These attacks are:

1. **Random** insertion attack, where random values replace original values for nc values in the series. This is done for c values in the range $[0.05, 0.1, 0.15, 0.2, 0.25, 0.3]$, representing 5% to 30% of the series length, reflecting varying levels of noise insertion.
2. **Shifting** the time series in time, forward or backward steps. This is done for x values in the range $[2, 4, 6, 8, 10, 12]$, representing shifts of up to 12 steps, which is a small percentage of the total series length, to simulate realistic time alignment issues.
3. **Offsetting**, or adding a constant value c to the entire series. This is done for values in the range $[0, \sigma]$ where σ is the standard deviation of the series, divided into 10 evenly spaced increments.
4. **Scaling** the entire series by a constant value c. This is done for values in the range $[0, \sigma]$ where σ is the standard deviation of the series, divided into 10 evenly spaced

increments.

5. **Min-Max** replacement of the k largest and smallest values in the time series by random values. The parameter chosen for k is 4 as it offers a sufficient amount of replacement to represent a realistic attack.

For all attacks, the resulting z-scores for each of the defined values are averaged to obtain a single representative result for each attack. This approach is reasonable as it accounts for the variability and potential impact of different magnitudes of modifications, providing a more comprehensive assessment of each watermarks' resilience to such attacks.

4 Experimental Setup and Results

4.1 Setup and Dataset

We use Chronos-T5-Base, a pretrained time series forecasting **model** with 200M parameters [3] to perform the main experiments and compare and analyse the performance of the watermarks. We also use the Lag-Llama model [21] to demonstrate the performance and cross-compatability of the watermarks over different models. In our results, the baseline values used for the time series are calculated as the mean values of the $n = 1000$ samples, given by the formula:

$$\text{Baseline value} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

Watermarking methods chosen for this experiment are KGW, EXP, and our proposed method the Heads Tails Watermark (`HTW`). For the `HTW` watermark, the parameter chosen for θ is 1. For the KGW watermark, we specifically apply the Soft-Red List variation with the recommended parameter of $\gamma = 0.25$, and a value of 4 for our modified δ parameter.

The context length, C, chosen for both the prediction quality and attack scenario experiments is 64 as this is the recommended value for both the Lag-Llama and Chronos models. The $seed$ used for both the pseudo-random function and all time series generation is 42.

The three **datasets** chosen for this study have varying frequencies (Monthly, Hourly, and Daily), quantities, and levels of complexity to ensure a robust evaluation of forecasting models. These datasets, detailed in Table 2, were chosen based on their relevance to common forecasting applications, their varying degrees of difficulty, and their representation of different domains. This diverse selection enables a thorough assessment of forecasting models' performance across different types of data, frequencies, and forecasting difficulties.

4.2 Prediction Quality

Please see Table 1 for the results over various prediction lengths, L. The results for the **Chronos** model reveal varying impacts on prediction quality across different datasets and lengths when applying the three watermarking techniques (`HTW` , KGW, and EXP).

1. For Air Passengers, `HTW` consistently shows the least deviation from the baseline sMAPE values, with minimal increases ranging from 0.02 to 0.04. This indicates that `HTW` has a negligible impact on prediction quality. In contrast, KGW displays more significant deviations, particularly at longer lengths, with increases up to 9.30. EXP shows moderate deviations, with an increase of up to 3.06, but generally lower than KGW.
2. In the Temperatures dataset, `HTW` again demonstrates minimal impact, with increases ranging from 1.96 to 3.79. KGW shows mixed performance, with slight decreases (up to -1.71) at some lengths and moderate increases at others, up to 1.70. EXP generally shows moderate increases, peaking at 5.13, indicating a slightly higher impact on prediction quality compared to `HTW` .
3. For Electricity, `HTW` exhibits no significant deviations, maintaining almost identical sMAPE values to the baseline (ranging from -0.00 to 0.00 when rounded to two decimals). KGW shows a more varied performance, with deviations ranging from -1.98 to 1.54, while EXP exhibits higher deviations, particularly at length 24 (5.78), suggesting a greater impact on prediction quality.

The results for the **Lag-Llama** model also show differences in the impact of watermarking techniques on prediction quality across datasets and lengths.

1. For Air Passengers, `HTW` shows a minimal to slightly negative impact, with changes ranging from -0.03 to 0.00, indicating a negligible effect. KGW shows significant deviations, particularly at longer lengths, with increases up to 58.89. EXP has the most substantial impact, with deviations ranging from 43.84 to 81.12, indicating a considerable reduction in prediction quality. The reason both EXP and KGW show such high deviations and bad performance for this model in the Air Passengers dataset could be due to a relatively high deviation in probability selection that the Lag-Llama model may have for this integer value-based dataset, resulting in increasingly bad performance when subsampling the probabilities as both watermarks do.
2. In the Temperatures dataset, `HTW` continues to maintain the least impact, with changes ranging from 0.06 to 1.17. KGW shows varied performance, with deviations ranging from 0.87 to 8.73. EXP again shows higher deviations, peaking at 11.89, indicating a moderate impact on prediction quality.
3. For Electricity, `HTW` shows negligible deviations, maintaining changes from -0.00 to 0.00. KGW demonstrates significant deviations, with values ranging from 3.22 to 16.40. EXP shows the highest deviations, particularly at longer lengths, with increases up to 33.89, indicating a substantial impact on prediction quality.

Overall, across both models, `HTW` exhibits minimal impact on prediction quality, showing the least deviation from baseline sMAPE values across all datasets and lengths with an average quality retention of 98.4%. KGW and EXP, however, show greater deviations, especially for the Lag-Llama model, where probability values have higher variance, with KGW performing variably and EXP generally showing moderate to significant impacts.

		Air Passengers				Temperatures				Electricity			
Model	**L**	**Base**	**ΔHTW**	**ΔKGW**	**ΔEXP**	**Base**	**ΔHTW**	**ΔKGW**	**ΔEXP**	**Base**	**ΔHTW**	**ΔKGW**	**ΔEXP**
Chronos	12	12.60	**0.04**	4.03	2.65	33.93	**3.79**	-1.71	5.13	0.95	**-0.00**	1.31	1.61
	24	20.59	**0.02**	4.54	0.57	34.43	**3.48**	-0.73	3.48	2.99	**0.00**	1.54	5.78
	36	27.27	**0.03**	2.95	0.92	31.15	**2.92**	1.59	1.33	6.39	**0.00**	-0.56	2.46
	48	32.14	**0.02**	-0.08	3.06	35.75	**2.43**	1.70	2.00	10.05	**-0.00**	-0.92	3.56
	60	37.01	**0.02**	0.84	0.07	36.68	**2.18**	1.52	1.71	11.24	**-0.00**	-1.98	2.96
	72	42.36	**0.03**	9.30	-0.43	37.47	**1.96**	1.52	1.97	11.03	**-0.00**	-1.79	4.19
Lag-Llama	12	18.76	**-0.03**	11.89	43.84	30.14	**1.17**	0.87	10.39	1.57	**0.00**	5.25	2.15
	24	19.27	**0.00**	27.28	59.09	29.36	**1.05**	6.70	9.06	4.77	**0.00**	3.22	13.90
	36	18.49	**-0.02**	31.32	71.46	26.08	**0.69**	2.14	11.89	6.00	**0.00**	14.55	21.25
	48	16.95	**-0.01**	43.87	79.81	29.12	**0.57**	8.73	9.94	9.58	**0.00**	10.46	30.62
	60	20.30	**-0.01**	58.89	81.12	27.91	**0.06**	7.48	8.25	12.57	**0.00**	16.40	30.22
	72	27.76	**-0.01**	36.95	75.11	27.86	**0.12**	6.65	7.37	15.09	**0.00**	14.32	33.89

Table 1: Differences from baseline sMAPE values for various prediction lengths and datasets for the Chronos and Lag-Llama models for $n = 1000$, $C = 64$ and $seed = 42$

Table 2: Datasets used in the research

Name	**Domain**	**Size**	**Frequency**
Air Passengers [7]	Passenger volume	144	Monthly
Electricity [12]	Electricity	230.736	Hourly
Melbourne [5]	Temperature	3.650	Daily

These results suggest that HTW is the most robust watermarking technique in terms of retaining prediction quality, while KGW and EXP may require careful consideration due to their higher impact on the time series values.

4.3 Detection Confidence and Robustness against Post-editing Attacks

We attacked each watermark with several transformations to assess how the confidence holds over different prediction lengths. The results of this can be found in Table 3. We exclusively used the Chronos model to perform this evaluation as it provides a comprehensive and consistent benchmark for robustness testing. By focusing on a single model, we were able to isolate the effects of the transformations more clearly and ensure that our assessment of robustness was not confounded by differences between models.

In analyzing the robustness of the watermarks under various attack scenarios, we compare the z-scores under attack conditions to their respective baseline z-scores. In the **baseline** scenario, the z-scores for all watermarks demonstrate varying degrees of robustness across different datasets. HTW consistently shows high scores (6.93 for all three datasets), indicating strong robustness. KGW also performs well, especially in the Electricity dataset where it significantly outperforms HTW with a z-score of 11.00. EXP, while demonstrating moderate robustness with scores around 4.18 to 4.79, is consistently lower than both HTW and KGW.

The **random** attack results in a reduction of z-scores for all watermarks compared to the baseline. KGW demonstrates a notable robustness with scores ranging from 4.33 for Air Passengers to 8.06 for Electricity. HTW 's performance is moderate, showing scores of 2.41 for Air Passengers and 5.00 for both Temperatures and Electricity, indicating good robustness for Temperatures and Electricity but poor for Air Passengers. These differing results are likely a result of variance, depending on whether or not the random attack managed to find and remove the minimum and maximum values in the series. EXP struggles under this attack with lower z-scores of 2.60 to 2.84, indicating poor robustness.

The **shift** attack shows significant variability in the z-scores. HTW maintains perfect z-scores of 6.93 across all datasets, indicating strong resilience due to the algorithm's reliance on only each value itself as seed for the random generation. In contrast, KGW's performance is highly variable, with scores ranging from a low of 0.11 for Air Passengers to a high of 3.78 for Temperatures. This can be explained as a result of in how far the attacks affected the seeding of each value. EXP's z-scores are close to zero across all datasets, highlighting its complete vulnerability to shift attacks due to its pseudo-random function depending on the c preceding values which can break under sufficiently large shift attacks.

The **offset** attack severely impacts the z-scores for KGW and EXP. KGW shows a dramatic decrease with a consistent z-score of -4.00 across all datasets, indicating a failure to withstand this attack. EXP also scores zero across all datasets. In contrast, HTW remains unaffected by the offset attack, maintaining its original baseline z-score of 6.93, demonstrating its strong robustness due to the normalization built in its algorithm.

Under the **scale** attack, the z-scores for HTW remain high due to normalization, with scores of 6.93 for Air Passengers and Temperatures, and slightly lower at 6.87 for Electricity, which could be the result of a floating-point rounding error. KGW, however, shows a consistent z-score of -4.00 across all datasets, indicating a significant vulnerability. EXP's z-scores are zero across the board, indicating no robustness to this type of attack.

The **min-max** attack results are mixed. HTW 's z-scores drop significantly, with scores of -0.58 for Air Passengers, -0.14 for Temperatures, and 0.29 for Electricity, indicating it fails to recognize the series. This is expected as the algo-

	Air Passengers			Temperatures			Electricity		
Attacks	**HTW**	**KGW**	**EXP**	**HTW**	**KGW**	**EXP**	**HTW**	**KGW**	**EXP**
Base	**6.93**	6.33	4.45	**6.93**	7.00	4.79	**6.93**	11.00	4.18
A_{random}	**2.41**	4.33	2.84	**5.00**	5.11	2.77	**5.00**	8.06	2.60
A_{shift}	**6.93**	0.11	0.00	**6.93**	3.78	0.02	**6.93**	-0.44	0.00
A_{offset}	**6.93**	-4.00	0.00	**6.93**	-4.00	0.00	**6.93**	-4.00	0.00
A_{scale}	**6.93**	-4.00	0.00	**6.93**	-4.00	0.00	**6.87**	-4.00	0.00
$A_{min-max}$	**-0.58**	6.17	3.38	**-0.14**	6.67	3.85	**0.29**	10.00	3.23

Table 3: z-score values for various datasets and attacks for the Chronos model for $C = 64$ and $L = 48$

rithm is highly dependent on the minimum and maximum values in the series, its main vulnerability. KGW performs well under this attack, showing z-scores of 6.17 for Air Passengers, 6.67 for Temperatures, and 10.00 for Electricity, indicating strong resilience. EXP demonstrates moderate robustness with scores ranging from 3.23 to 3.85 across all datasets.

The analysis reveals that HTW generally exhibits strong robustness across most attacks with an average z-score of 5.28, except for the min-max attack where its performance significantly drops, which is expected due to the design of the algorithm and its dependence on the minimum and maximum values of the series for normalization. KGW shows exceptional performance in the baseline and min-max attack, particularly for the Electricity dataset, but is highly vulnerable to shift, offset, and scale attacks. EXP, while maintaining moderate scores, fails to show strong resilience under most attacks, particularly the shift, offset, and scale attacks. These results thus demonstrate the weakness of conventional Large Language Model watermarks and the strengths of the specialised HTW to such post-processing time series attacks.

5 Responsible Research

5.1 Ethical Considerations

Algorithmic transparency has become a significant concern in recent times as we seek to improve trust in machine learning systems and prevent issues such as algorithmic bias [27]. This is especially important due to the increasing pervasiveness of machine-generated decision making in our day-to-day lives. That is why we have published and explained every step of our algorithm in this paper.

Potential future iterative improvements on the algorithm will be based solely on experimental results and will be transparently published to ensure no changes will be unclear.

Misuse and incorrect usage of the algorithm can possibly lead to incorrect conclusions about whether or not a time series was machine-generated. The algorithm presented in this paper should thus solely be used for experimental and exploratory purposes.

5.2 Privacy

Datasets used for this research are all publicly available. No personal or private data has been gathered or used during the experiments, nor is there any data that can be traced to individual people.

Watermarking as a method can result in privacy concerns due to the ability to attribute a time series to a specified origin. We want to emphasize that the algorithm in this paper is not intended to be used for commercial or governmental use and only for research purposes.

5.3 Reproducibility of Results

All algorithms used have a limited deterministic randomness in their operations. Thus, to allow for full reproducibility of methods and results, we have disclosed all seeds and runtime configurations as clearly as possible in this paper.

6 Conclusions and Future Work

We introduce HTW , a novel method for watermarking time series foundation models, providing a different approach from probabilistic watermarking methods that rely on a series' distribution for detection and insertion. Through comprehensive evaluation on two foundation models and three datasets, we demonstrate that the proposed HTW watermark is a viable alternative to these methods in benchmark performances such as prediction quality and watermark detection. Specifically, throughout our experiments, HTW maintains an average prediction quality retention of 98.4% and achieves an average z-score of 5.28 over several attack scenarios for a prediction length of 48. These experiments thus show that HTW not only exceeds traditional watermarks in almost every metric but that it is able to consistently deliver high prediction quality and detection accuracy.

Future work could look into the effect different training sizes and context lengths have on the performance of the watermarks. Another possible point of research is the application of a watermark to other aspects of a time series than the numeric structure and probabilistic range such as the frequency.

References

[1] Scott Aaronson. Watermarking of large language models, 2023. Accessed: 2024-06-01.

[2] Sahar Abdelnabi and Mario Fritz. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. *Arxiv*, September 2020.

[3] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series, 2024. Accessed: 2024-06-01.

[4] Yodi Assi, Casten Baum, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. *Usenix*, August 2018.

[5] Australian Bureau of Meteorology. Daily minimum temperatures in melbourne, australia, 1990. Available from R 'datasets' package.

[6] Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986.

[7] George EP Box, Gwilym M Jenkins, and Gregory C Reinsel. *Time Series Analysis: Forecasting and Control*. Holden-Day, 1976.

[8] Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. Cryptology ePrint Archive, Paper 2023/763, 2023. https://eprint.iacr.org/2023/763.

[9] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

[10] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting, 2023.

[11] Azul Garza and Max Mergenthaler-Canseco. Timegpt-1, 2023.

[12] R. Godahewa, C. Bergmeir, G. Webb, R. Hyndman, and P. Montero-Manso. Australian electricity demand dataset (version 1), 2021. Data set.

[13] Alexei Grinbaum and Lukas Adomaitis. The ethical need for watermarks in machine-generated language. *arXiv preprint arXiv:2209.03118*, 2022.

[14] Rob J Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. *Forecasting with Exponential Smoothing: The State Space Approach*. Springer Science & Business Media, 2008.

[15] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *Proceedings of The Twelfth International Conference on Learning Representations*, 2024.

[16] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. *Arxiv*, june 2023.

[17] Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models, 2023. Accessed: 2024-06-01.

[18] Jiacheng Ma, Pablo Montero-Manso, Rob Hyndman, Geoffrey Webb, and Christoph Bergmeir. Large language models are zero-shot time series forecasters. *arXiv preprint arXiv:2310.07820*, 2023.

[19] Suvir Mirchandani, Fei Xia, Pete Florence, Brian Ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. Large language models as general pattern machines. In *Proceedings of The 7th Conference on Robot Learning*, volume 229, pages 2498–2518. PMLR, 2023.

[20] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[21] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-llama: Towards foundation models for probabilistic time series forecasting. *arXiv preprint arXiv:2310.08278*, 2023.

[22] Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8857–8868. PMLR, July 2021.

[23] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. In *International Journal of Forecasting*, volume 36, pages 1181–1191. Elsevier, 2020.

[24] Neeraj Kumar Sharma, Subodh Kumar, and Naveen Kumar. Hgsmark: An efficient ecg watermarking scheme using hunger games search and bayesian regularization bpnn. *Biomedical Signal Processing and Control*, 83:104633, 2023.

[25] P. V. Sai Charan Shukla et al. From text to mitre techniques: Exploring the malicious use of large language models for generating cyber attack payloads. *arXiv preprint arXiv:2305.15336*, 2023.

[26] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bash-

lykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[27] Harriet Webb, M. Patel, Michael Rovatsos, A. Davoust, S. Ceppi, Ansgar Koene, L. Dowthwaite, V. Portillo, M. Jirotka, and M. Cano. "it would be pretty immoral to choose a random algorithm": Opening up algorithmic interpretability and transparency. *Journal of Information, Communication and Ethics in Society*, 17(2):210–228, 2019.

[28] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers, 2024.

[29] Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained lm. In *Advances in Neural Information Processing Systems*, 2023.