

# Dataset Watermarking Using the Discrete Wavelet Transform

Mike P. Raave<sup>1</sup><sup>a</sup>, Devriş İşler<sup>2,3</sup><sup>b</sup> and Zekeriya Erkin<sup>1</sup><sup>c</sup>

<sup>1</sup>Cyber Security Group, Delft University of Technology, Van Mourik Broekmanweg, Delft, The Netherlands

<sup>2</sup>IMDEA Networks Institute, Madrid, Spain

<sup>3</sup>Universidad Carlos III de Madrid, Madrid, Spain

**Keywords:** Watermarking, Data Ownership, Discrete Wavelet Transform.

**Abstract:** In this work, we focus on watermarking time series datasets and explore one of the techniques known from audio-watermarking, namely Discrete Wavelet Transform (DWT) based watermarking, to investigate its effectiveness. We adapt (Attari and A. Shirazi, 2018) and embed a bit stream into a time series dataset by calculating the DWT coefficients and modifying their magnitudes for embedding. Our experimental results on two real-world datasets show good robustness against a small range of data modification attacks but lack capability in larger-scale attacks. We believe that our work could initiate a new research direction on dataset watermarking using well-known techniques from signal processing.

## 1 INTRODUCTION


Watermarking is a well-known technique for protecting data ownership upon unauthorized distribution. Watermarking generally consists of two main algorithms: 1) *embedding*; and 2) *extraction*. Embedding allows an owner to embed a watermark into a form of data using a watermarking secret (e.g., a secret bit string) and produces a watermarked version of the data without degrading the data utility significantly (e.g., minimizing the amount of distortion on median and average). In extraction, the owner proves its ownership of a suspected dataset, even if the data is modified, by reconstructing, or extracting, the embedded watermark. When the watermark is successfully extracted, the owner can prove the ownership by verifying their watermark.


Watermarking has been well-studied in the context of multimedia data (Malanowska et al., 2024), database (Rani and Halder, 2022; Panah et al., 2016), and other type of datasets (İşler et al., 2024). Although media watermarking, e.g. image, audio, is more advanced than non-media watermarking, there are only a few studies analyzing the implications of media watermarking on non-media real-world datasets. For example, (Pham et al., 2017) attempt to


watermark medical time-series datasets using DWT-based watermarking. They deploy an ML model to retrieve a watermark formed as a binary image embedded into the dataset. Due to its use case, i.e., more effective diagnoses, the reconstruction of medical signals is vital. Therefore, their approach is not directly applicable to other time series applications. (Maesen et al., 2023), on the other hand, watermark machine learning datasets by applying singular value decomposition based image watermarking which cause around 0.1% accuracy loss. However, their approach is not generic and cannot be directly applied to the datasets such as time-series.

Considering time-series and audio data, they have common characteristics such as both data types are a form of frequency and are measured over time. Therefore, in this work, we explore the idea of applying a transform domain media watermarking algorithm, i.e., DWT-based audio watermarking, to non-medical time series data. Transform domain algorithms are more robust and resistant to attacks (Guo et al., 2023; Attari and A. Shirazi, 2018) while ensuring better imperceptibility compared to spatial domain algorithm, e.g., Least Significant Bit. In consequence, we chose DWT-based audio watermarking (Attari and A. Shirazi, 2018) since it is a frequency domain watermarking algorithm and makes good use of the signal characteristics (Guo et al., 2023).

**Our Approach.** To be able to watermark a time-series dataset that may contain multiple signals and

<sup>a</sup> <https://orcid.org/0009-0003-0751-3773>

<sup>b</sup> <https://orcid.org/0000-0003-4895-8827>

<sup>c</sup> <https://orcid.org/0000-0001-8932-4703>

several covariates (which is not the case for media signals) using DWT-based audio watermarking (Attari and A. Shirazi, 2018), we modified their algorithm to suit the type and range of time-series data. We split the data into frames and embed a Barker code into each frame to counter single-range cropping attacks and enhance performance. Afterwards, we use the magnitude of the DWT coefficients of the data to embed the watermark. Additionally, we proposed new attacks to improve the robustness of the modified algorithm after careful investigation since the original algorithm's use-case scenario is changed.

Our foremost contribution is to apply an audio watermarking technique based on DWT to non-media time series datasets using two real-world datasets. We measure the distortion in terms of data statistics introduced by watermarking. Based on our experimental finding, our algorithm modifies the average of the datasets by approximately 0.5%. Hence, adjusting the watermarking parameters, the owner can satisfy its desired imperceptibility and robustness. Later, we evaluate robustness against *cropping*, *scaling*, *noise addition*, *zero-out*, and *collusion* attacks. Our experimental results show that our algorithm improves robustness up to 65% data deletion for single-range cropping attacks and up to a scaling rate of 1.5 for scaling attacks.

## 2 RELATED WORK

(Attari and A. Shirazi, 2018) is an SS-based audio watermarking approach. They watermark an audio signal using 6<sup>th</sup>-level DWT coefficients. These coefficients are divided into frames, after which a watermark bit is assigned per frame. Per coefficient in a frame, the closest Fibonacci number to its magnitude is determined. Based on the bit assigned to the frame, which contains the coefficient, and the Fibonacci number, the magnitude of the coefficient is changed to either the closest Fibonacci number or one number higher in the Fibonacci sequence. To extract the watermark, similar steps as in the embedding process are executed frame by frame. If, within a frame, the majority of the closest Fibonacci numbers to the coefficients correspond to even positions in the Fibonacci sequence, the extracted watermark bit is 0, otherwise, it is 1. As the bit stream is randomly generated, the algorithm is hard to break ensuring a high level of security. They further analyze the robustness of their algorithm against various attacks. In addition, their algorithm is suitable for various applications since the frame size can be tuned depending on the underlying watermarking application.

(Fallahpour and Megías, 2014) propose a similar algorithm utilizing the Fast Fourier Transform (FFT) coefficients instead of DWT coefficients and gives the possibility to tune parameters (e.g., the frequency band of the signal used for embedding and the frame size) depending on the required capacity and robustness. As FFT coefficients do not store information about time, their algorithm is only suitable for audio applications. (Pham et al., 2017) use an image as a watermark that is scrambled through the Arnold transformation before embedding to improve robustness. The data itself is down-sampled after the 4<sup>th</sup>-level DWT approximation coefficients are calculated. Afterwards, the watermark image gets embedded with a bit of the image per data frame. For extraction, they use an ML model built from part of the watermarked data. This algorithm has high robustness against small forms of cropping, noise addition, filtering and re-sampling attacks and has good imperceptibility such that the watermarked data is still usable.

## 3 WATERMARKING TIME SERIES USING DWT

Now, we present our watermarking scheme utilizing (Attari and A. Shirazi, 2018) to watermark time-series.

### 3.1 Watermark Embedding

The watermark embedding algorithm, see Algorithm 1, receives the following inputs: a time-series dataset  $d$  formalized as  $d = \{(t_i, x_i) | i = 1, 2, \dots, N\}$  where  $N$  is the total number of observations in  $d$ , and  $t_i$  is the timestamp of the  $i$ -th observation and  $x_i \in \mathbb{R}$  is the  $i$ -th observation's value(s),<sup>1</sup> a bit stream as watermark  $w$ , a reference set  $ref$  of numbers with a fixed multiplication rate  $mult$ , an integer containing the preferred DWT level  $dwt$ , a frame size for the coefficients  $f_{bin}$ , a frame size for embedding a Barker code  $f_{Barker}$ , and returns a watermarked version of  $d$  as denoted by  $d_w$ .  $w$  is randomly generated through a pseudorandom function using a secret seed value as a high entropy secret.  $d$  is split into smaller sections such that  $w$  is embedded into multiple places in  $d$  to increase robustness against (un)intentional modifications. Each section  $n$  has a fixed size of  $f_{Barker}$ . Per section  $n$ , the data is split into two lists,  $n_1$  and  $n_2$ , such that  $n_1 + n_2 = n$  as in (Shen et al., 2012).

<sup>1</sup>The formalization can be modified as  $x_i \in \mathbb{R}^m$  where  $m \geq 1$  represents the number of variables or dimensions (univariate if  $m = 1$ , multivariate if  $m > 1$ ).

In  $n_1$  5-bit Barker codes are embedded in each sequence to counter cropping attacks. The relative value of a Barker code of length 5 has been used as considered in (Campbell, 1989) which is added to the value at the corresponding index in  $n_1$ . The imaginary values of the Barker code are added in the dataset to make the process of extracting the watermark more robust and effective. In order to avoid having imaginary values in the dataset, we use  $n_1$  for the Barker code and  $n_2$  for embedding the watermark. For em-

in  $ref$ . After all coefficients are considered, the inverse DWT is applied and  $n_1$  is merged with the modified version of  $n_2$  to get the watermarked data  $d_w$ . Finally, all the sections are concatenated to form  $d_w$ , and  $\{f_{bin}, f_{Barker}, dwt, ref, w\}$  are securely stored as watermark secrets.

A few steps in the original algorithm are modified to improve the performance. First, splitting the data into lists with size  $f_{Barker}$  and dividing those lists  $n$  into  $n_1$  and  $n_2$  is an improvement to accommodate the embedding of a 5-bit Barker code in  $n_1$  to counter single-range cropping attacks. Second, this algorithm has no variable values which are optimal for all cases compared to the original. Thus, the user/owner can determine the way the variable  $ref$  is created and what the value of  $f_{bin}$ ,  $f_{Barker}$  and  $dwt$  is. The results of changing these variables can be seen in Section 4.

Algorithm 1: Watermark Embedding.

---

**Data:**  $d, f_{bin}, f_{Barker}, dwt, ref, w$   
**Result:**  $d_w$

```

1 foreach List  $n$  from  $d$  with size  $f_{Barker}$  do
2    $n_1 = [1, \dots, 1]_{|f_{Barker}|}$ ;
3    $n_2 = n - n_1$ ;
4    $n_1[1] = 1 + \sqrt{-1}$ ;
5    $n_1[3] = 1 - \sqrt{-1}$ ;
6    $c_A = \text{ComputeDWT}(n_2, dwt)$ ;
7   for List  $y$  in  $c_A$  with size  $f_{bin}$  do
8      $frameNr =$  the current iteration of the
      loop.;
9      $w_{bit} = w[frameNr]$ ;
10    for Coefficient  $c$  in  $y$  do
11       $k =$ 
         $findPositionClosestNumber(ref, c)$ ;
12      if  $k \bmod 2 \equiv w_{bit}$  then
13         $c = ref[k]$ ;
14      end
15      else
16         $c = ref[k + 1]$ ;
17      end
18    end
19  end
20   $n'_2 = \text{ComputeIDWT}(c_A)$ ;
21   $n = n_1 + n'_2$ ;
22 end
23  $d_w = \text{CombineAll}(n)$ ;

```

---

bedding the watermark in  $n_2$ , a  $dwt^{th}$ -level DWT is first applied to  $n_2$  and the resulting DWT approximation coefficients are divided into frames  $f$  of size  $f_{bin}$ . Each frame is assigned a bit out of  $w$  by calculating the index of the bit in  $w$ :  $l = \lfloor \frac{i}{index\ of\ f} \rfloor + 1$ , where  $i$  represents the index of the coefficient in the list of approximation coefficients. For each coefficient  $c$ , the  $k^{th}$  number in  $ref$  is found that is closest to the magnitude of  $c$ . To embed the bit, the condition  $k \bmod 2 \equiv w[l]$  is checked. If the condition is true, the magnitude is changed to the  $k^{th}$  number in  $ref$ . If not, it is changed to the  $(k + 1)^{th}$  number

Algorithm 2: Watermark Extraction.

---

**Data:**  $d_w, f_{bin}, f_{Barker}, dwt, ref$   
**Result:**  $w'$

```

1  $w' = []$ ;
2  $startIndex = detectBarkerCode(d_w, f_{Barker})$ ;
3 Take a sub-list with the first  $f_{Barker}$  values
  from  $startIndex$  as  $n$ ;
4  $n_1 = [1, \dots, 1]_{|f_{Barker}|}$ ;
5  $n_2 = n - n_1$ ;
6  $c_A = \text{ComputeDWT}(n_2, dwt)$ ;
7 for List  $y$  in  $c_A$  with size  $f_{bin}$  do
8   for Coefficient  $c$  in  $y$  do
9      $k = findPositionClosestNumber(ref,$ 
       $c)$ ;
10    Store  $k \bmod 2$ ;
11    if majority- $k$ -values  $\equiv 0$  then
12       $w'.add(0)$ ;
13    end
14    else
15       $w'.add(1)$ ;
16    end
17  end
18 end
19 return  $w'$ ;

```

---

### 3.2 Watermark Extraction

The first step in watermark extraction, see Algorithm 2, is to detect the start point of extracting the watermark from a suspected dataset  $d_w$  using  $\langle f_{bin}, f_{Barker}, dwt, ref \rangle$ . The detection is achieved by finding the embedded Barker codes at the start of each section, which are the added imaginary values to the data. When the start of a section has been found, a list  $n$  is stored from the start index with the next  $f_{Barker}$

values. This list is split into sections  $n_1$  and  $n_2$ , such that  $n_1 + n_2 = n$ . A  $dwt^{th}$ -level DWT is applied to  $n_2$  and the resulting DWT approximation coefficients are divided into frames with size  $f_{bin}$ . Each frame  $y$  is used for a form of majority voting to find the watermark bit embedded. Hence, for each coefficient  $c$  in a frame, the  $k^{th}$  number is found in  $ref$  closest to the magnitude of  $c$ . Each  $k$  connected to each  $c$  is stored. If the values of  $k$  are more often odd than even per frame, the watermark bit is 1; otherwise, it is 0. Therefore, the algorithm always uses odd-sized frames to prevent ambiguity. Once all frames have been considered, the watermark  $w'$  is extracted from the sequence. If  $w'$  is equal to the input watermark  $w$ , the watermark is successfully detected.

The main change to the extraction algorithm compared to the original is the added step to detect the Barker code at the start of a section. Since existing methods offered no clear way to embed and detect Barker codes, we used complex values, which remain hidden when the data is visualized. Additionally, the algorithm only analyzes the first section from the starting point, as it offers the highest likelihood of correct watermark retrieval, especially when the dataset has been modified or resized.

## 4 EXPERIMENTAL SETUP AND RESULTS

### 4.1 Setup

Our experimental results are produced on a standard laptop machine with an Intel(R) Core(TM) i7-9750H CPU at 2.6GHz with 16.00 GB RAM. We use two datasets from (Brownlee, 2020): 1) minimal temperature dataset; and 2) sunspots dataset. The sunspots dataset and the minimal temperature dataset consist of 2800 entries and 3600 entries, respectively. In terms of the statistics of the datasets, the sunspots dataset has an average value of 51.3. The minimal temperature dataset has an average value of 11.2. For our experiments, we evaluate our approach using two main metrics: 1) *imperceptibility*; and 2) *robustness*.

*Imperceptibility.* As a common practice (Agrawal and Kiernan, 2002), we evaluate the imperceptibility by analyzing the effect of the watermark on each dataset by measuring the relative changes of the following metrics: 1) the average of the dataset; and 2) the average absolute change of values.

*Robustness.* We evaluate the robustness of our approach against five attacks: 1) *cropping*; 2) *noise addition*; 3) *scaling*; 4) *zero-out* attacks; and 5) *collusion* attacks. These particular attacks have been cho-

Table 1: Imperceptibility results of the test datasets.

Input			Min temp (%)		Sunspots (%)	
$f_{bin}$	dwt	mult	avg diff	value chg	avg diff	value chg
11	1	1.2	0.5	8.9	0.7	11.9
11	3	1.2	0.4	8.8	0.6	11.9
11	5	1.2	0.8	8.9	1.0	19.4
11	3	1.4	1.0	16.1	2.3	22.7
15	1	1.2	0.5	8.9	0.7	11.9

sen since we focus on formatted datasets. As a result, these datasets would be most vulnerable to standard data modification attacks as the ones mentioned above. In the original algorithm (Attari and A. Shirazi, 2018), there are results for a form of cropping and random noise addition attack which we can compare with this algorithm. For the other three attacks there are no results to compare with from the original algorithm as these attacks are more typical for time series data compared to audio data. Regarding the value of  $f_{Barker}$ , it was found that the smaller the value is, the more perceptible the watermark will be as a change in DWT values in a small piece of data has a larger effect on the resulting value than with a larger value of  $f_{Barker}$ . The larger the value is, the less robust the watermark is as a smaller part of the data needs to be modified for the watermark to be irretrievable. The only constraint for  $f_{Barker}$  is that the value must be smaller or equal to the size of the dataset. However, we have run all tests with  $f_{Barker} = 600$ , as this provided good imperceptibility and did not have any negative effects on the robustness of the algorithm compared to lower values of  $f_{Barker}$ .

### 4.2 Results

We discuss the experimental results based on the aforementioned setting and metrics. We run our experiments 100 times per instance and take the mean of total computations.

#### 4.2.1 Imperceptibility

We present the complete results of imperceptibility experiments in Table 1 and briefly discuss the results due to limited space. The complete results can be found in the full version (Raave et al., 2024).

*Effect of  $f_{bin}$ .* Our results show that when we increase  $f_{bin}$ , the effect on the statistics of the datasets are not significant as the average difference and average value do not change for both datasets. Thus, our approach does not distort data drastically.

*Effect of dwt.* When  $dwt$  is increased, we can see in the results that the relative difference in the av-



erage value and the average change per value in the dataset increases noticeably when  $dwt = 5$  for all three datasets. Between  $dwt = 1$  and  $dwt = 3$  each dataset does not show significant differences, which shows that both values should achieve the best possible results when chosen in the algorithm.

*Effect of mult.* When  $mult$  is increased, all of the resulting values increase with a significant factor. This effect can be observed in both datasets.

#### 4.2.2 Robustness

For the baseline for our robustness analysis, we chose the following values:  $f_{bin} = 11$ ,  $dwt = 1$  and  $mult = 1.2$ , as those values gave the most consistently good results regarding watermark extraction and provided a good trade-off between imperceptibility and robustness. However, optimizing these parameters is an intriguing open problem as we did not include more values of  $f_{bin}$ ,  $dwt$ , and  $mult$ , since the detection rate of the watermark would drop below a 100% when the values deviated too far from the set baseline. For robustness, the following data modification attacks have been chosen to focus on: cropping, noise addition, scaling, zero-out, collusion attacks. These attacks were tested with 10%, 30% and 50% of the dataset being modified. These modifications were done on randomly chosen indexes. Higher than 50% modification led to a conversion of a Bit Error Rate (BER) of 50%. As the algorithm only considers zeros and ones as watermarks, it cannot be distinguished from that point whether the algorithm randomly guesses, as the correct guess for a watermark bit is 50%, or the algorithm makes more mistakes for a different reason. Below we discuss the results per attack.

Table 2: Robustness results of the real world datasets.

	Min temp (%)				Sunspots (%)			
	Crop	Edit	Insert	Zero	Crop	Edit	Insert	Zero
10	35	10	45	10	35	15	40	15
30	50	45	50	40	50	40	50	40
50	50	50	50	45	50	50	50	50

**Cropping Attack.** Two types of cropping attacks were considered: removing a continuous range of data and randomly deleting samples. To resist these, the algorithm uses Barker codes for synchronization. By choosing a frame size  $f_{Barker}$ , the watermark is embedded in each frame, increasing the chance of recovery. A value of  $f_{Barker} = 600$  (for datasets of  $\sim 3000$  entries) offers a good balance between robustness and imperceptibility. With this setup, up to 65% of the data can be removed in one continuous block, and

the watermark is still fully recoverable. However, the method is less effective against random deletion. As shown in Table 2, removing 10% of samples at random results in a BER of 35%, only slightly better than random guessing (50%).

**Random Noise Addition Attack.** For adding noise to the data, two variants of this attack are chosen to test with: editing the existing values and inserting new entries into the dataset. For editing existing values the edited values were restricted to be within the range of 10% of the current value so that it is less perceptible to others that the data is modified as shown by Table 2. Table 2 shows that for both datasets, the BER is relatively low when 10% of the data is modified. For larger-scaled attacks, it is more vulnerable.

For inserting values, they were chosen in a similar way as for editing. The results of these tests can be found in Table 2. In the table, it can be seen that for both datasets, the watermark cannot be retrieved reliably anymore. For 10% data modification, the BER is lower than when one would guess the watermark bits, but not significantly lower such that one could still use the watermarked data in a practical setting.

**Zero-Out Attack.** A zero-out attack entails that a percentage of the data is changed to have a value of zero. Therefore it is similar to a cropping attack, but the dataset does not shrink in size in this case. Table 2 shows the results of the experiments. As shown by Table 2, we observe that for 10% of the datasets being modified, the algorithm has a BER of 15%. With larger-scaled zero-out attacks, this algorithm proves to be more vulnerable.

**Scaling Attack.** A scaling attack entails that the data is being scaled with a certain factor. The results of this attack being successful were fully dependent on the value of  $mult$ . The tests have been run with multiple values of  $mult$ , but 1.6 seemed to give the best results. We found in our results that for positive scaling values, the algorithm provides strong robustness up to 50% scaling with a resulting BER of 0%. When the data is negatively scaled, the algorithm seems to flip all bits which results in a BER of 100%. When testing with lower values of  $mult$ , the maximal scaling rate which still resulted in a BER of 0% detection went down to about 10% positive scaling. When comparing these results to a medical time series watermark in (Pham et al., 2017) and the audio algorithm (Attari and A. Shirazi, 2018), our algorithm provides more robustness in positive scaling, while being less robust against negative scaling (see our full version (Raave et al., 2024) for more details).

**Collusion Attack.** In a collusion attack, an attacker has access to two or more differently watermarked versions of the same dataset and attempts to remove

or obscure the watermarks. This attack assumes a stronger adversary than other types. To test our modified algorithm's robustness, we generated two versions of the same dataset with different watermarks and parameters (e.g.,  $f_{Barker}$ ,  $dwt$ ,  $mult$ ). Our results show that recovering the watermark or parameters is difficult, as each one affects the data in a random, uncorrelated way. Additionally, the use of a high-entropy secret in generating  $w$  prevents the watermark from being guessed. We will explore more advanced collusion scenarios involving multiple copies.

## 5 DISCUSSION

Our imperceptibility tests show that changing  $f_{bin}$  and DWT level has minimal impact on output quality, suggesting that a fixed DWT level is not essential for time series data as it is common in audio watermarking. For cropping attacks, the use of Barker codes proved partially effective: with  $f_{Barker} = 600$ , only one intact block of 600 values is needed to fully recover the watermark. This outperforms the method in (Attari and A. Shirazi, 2018), which handles only a 200-sample removal at the beginning of an audio signal. While our method remains robust when up to 10% of data is affected at random, performance drops beyond that.

Unlike in Attari et al.'s audio watermarking, using a Fibonacci  $mult$  of 1.618 in time series data increased perceptibility and had limited robustness benefits. A smaller  $mult$  (1.2) worked better, likely due to the lower value spread in time series compared to audio. While we use standard distortion metrics, distortion is highly use-case dependent and influenced by attack types. As noted in (İşler et al., 2024), optimizing and generalizing distortion remains a complex challenge, which we plan to explore further both theoretically and experimentally.

## 6 FUTURE WORK AND CONCLUSION

We proposed a novel technique to watermark non-medical time series by adapting an audio watermarking technique (Attari and A. Shirazi, 2018). We experimentally showed that our approach is robust against *cropping*, *scaling*, and small-scale *randomly sampled* attacks. We also evaluated the effect of watermarking on data utility (imperceptibility) using different multiplication rates. Determining the optimal parameters for a given time-series dataset is an interesting future direction.

## ACKNOWLEDGEMENTS

This paper is supported by the European Union's Horizon Europe research and innovation program under grant agreement No. 101094901, the Septon and 101168490, the Recitals Projects. Devriş İşler was supported by the European Union's HORIZON project DataBri-X (101070069).

## REFERENCES

- Agrawal, R. and Kiernan, J. (2002). Watermarking relational databases. In *VLDB*. Morgan Kaufmann.
- Attari, A. and A. Shirazi (2018). Robust audio watermarking algorithm based on DWT using fibonacci numbers. *Multim. Tools Appl.*
- Brownlee, J. (2020). 7 time series datasets for machine learning.
- Campbell, C. (1989). 13 - coding techniques using linear saw transducers. In *Surface Acoustic Wave Devices and their Signal Processing Applications*. Academic Press.
- Fallahpour, M. and Megías, D. (2014). Robust audio watermarking based on fibonacci numbers. In *MSN*.
- Guo, K., Xu, Z., Luo, S., Wei, F., Wang, Y., and Zhang, Y. (2023). Invisible video watermark method based on maximum voting and probabilistic superposition. In *ACM MM*.
- İşler, D., Cabana, E., García-Recuero, Á., Koutrika, G., and Laoutaris, N. (2024). Freqywm: Frequency watermarking for the new data economy. *IEEE ICDE*.
- Maesen, P., İşler, D., Laoutaris, N., and Erkin, Z. (2023). Image watermarking for machine learning datasets. In *ACM Data Economy Workshop, DEC*.
- Malanowska, A., Mazurczyk, W., Araghi, T. K., Megías, D., and Kuribayashi, M. (2024). Digital watermarking - A meta-survey and techniques for fake news detection. *IEEE Access*.
- Panah, A. S., van Schyndel, R. G., Sellis, T. K., and Bertino, E. (2016). On the properties of non-media digital watermarking: A review of state of the art techniques. *IEEE Access*.
- Pham, T. D., Tran, D., and Ma, W. (2017). An intelligent learning-based watermarking scheme for outsourced biomedical time series data. In *IJCNN*.
- Raave, M. P., İşler, D., and Erkin, Z. (2024). Watermarking time-series data using dwt. *TU Delft repository*.
- Rani, S. and Halder, R. (2022). Comparative analysis of relational database watermarking techniques: An empirical study. *IEEE Access*.
- Shen, J., Pan, F., and Guo, Y. (2012). Digital audio watermark sharing based on the chinese remainder theorem. In *International Congress on Image and Signal Processing*.