SpeechVerifier: Robust Acoustic Fingerprint against Tampering Attacks via Watermarking

Lingfeng Yao[1*] **Chenpei Huang**[1*] **Shengyao Wang**[2] **Junpei Xue**[2]
Hanqing Guo[3] **Jiang Liu**[2] **Xun Chen**[4] **Miao Pan**[1]
[1]University of Houston
[2]Waseda University
[3]University of Hawaii at Mānoa
[4]Independent Researcher

Abstract

With the surge of social media, maliciously tampered public speeches, especially those from influential figures, have seriously affected social stability and public trust. Existing speech tampering detection methods remain insufficient: they either rely on external reference data or fail to be both sensitive to attacks and robust to benign operations, such as compression and resampling. To tackle these challenges, we introduce SpeechVerifer to proactively verify speech integrity using only the published speech itself, i.e., without requiring any external references. Inspired by audio fingerprinting and watermarking, SpeechVerifier can (i) effectively detect tampering attacks, (ii) be robust to benign operations and (iii) verify the integrity only based on published speeches. Briefly, SpeechVerifier utilizes multiscale feature extraction to capture speech features across different temporal resolutions. Then, it employs contrastive learning to generate fingerprints that can detect modifications at varying granularities. These fingerprints are designed to be robust to benign operations, but exhibit significant changes when malicious tampering occurs. To enable speech verification in a self-contained manner, the generated fingerprints are then embedded into the speech signal by segment-wise watermarking. Without external references, SpeechVerifier can retrieve the fingerprint from the published audio and check it with the embedded watermark to verify the integrity of the speech. Extensive experimental results demonstrate that the proposed SpeechVerifier is effective in detecting tampering attacks and robust to benign operations.

1 Introduction

Audio serves as an important information carrier that is widely used in news reporting, legal evidence, and public statements. However, the rapid development of audio editing tools [30] and text-to-speech (TTS) generation models [16, 20, 31, 9] has significantly lowered the technical barriers for speech manipulation and synthesis. While these techniques benefit content creation and entertainment, they also enable attackers to tamper speech content with ease. Public speeches and statements, especially made by influential figures, have become prime targets for attacks due to their huge social impact. Tampered speech can cause the spread of misinformation, undermine public trust, and even threaten social stability. Moreover, the prevalence of social media platforms accelerates the circulation of tampered audio, posing challenges to ordinary people in identifying the authenticity from numerous sources. For instance, some statements of U.S. Presidents have been repeatedly edited and manipulated, and broadly redistributed on various social media platforms across the internet [23, 21]. As such doctored speeches reach the public, doubt naturally arises concerning their

[*]Equal contribution.

Preprint.

integrity. Currently, verifying the truth often requires cross-checking information across multiple social media platforms, a process that is time-consuming and prolongs the spread of misinformation. These challenges highlight a critical need: Is it possible to proactively protect publicly shared speech against tampering attacks while still allowing it to be freely stored, distributed, and reshared?

Existing approaches against speech tampering attacks can roughly be categorized into two groups: passive detection and active protection. Passive detection methods [25, 33, 18, 12, 3] primarily rely on deep binary classifiers trained to identify subtle artifacts introduced by tampering operations. While they show reasonable performance against known attacks, their sensitivity to unseen or sophisticated manipulations remains limited. Moreover, passive detection alone cannot verify whether the speech content originates from the claimed speaker, leaving systems vulnerable to impersonation-based attacks [11]. On the other hand, active protection methods aim to ensure content integrity by embedding auxiliary information into the audio signal or extracting it during verification. Common approaches include cryptographic hashing [27] and fragile watermarking [22], both of which can reliably detect content alterations. However, these methods are highly sensitive to benign operations such as compression or resampling, restricting their applicability in real-world distribution scenarios. Furthermore, hash-based verification typically requires transmitting or retrieving external reference hash values, limiting their ability for independent self-verification of the published speech audio.

To address the issues above, a desired speech verification design should have the following properties: (1) **Convenient to use**: the integrity of the speech can easily be verified by the general public without requiring external references. (2) **Sensitive to tampering attacks**: it can reliably detect any malicious edits, including subtle semantic (e.g., can $\Leftrightarrow$ cannot) or speaker-related (e.g., timbre) changes. (3) **Robust to benign operations**: it should be robust to typical benign audio operations, especially commercial-off-the-shelf codecs (e.g., AAC or Opus in Instagram/TikTok), ensuring usability in sharing and distribution. Therefore, in this paper, we propose SpeechVerifier, a proactive acoustic fingerprint-based speech verification design that jointly utilizes semantic content and speaker identity. Specifically, SpeechVerifier uses multiscale feature extraction to capture speech features across different temporal resolutions. Then, it employs contrastive learning to generate fingerprints that can detect modifications at varying granularities. These fingerprints are designed to be robust to benign operations, but exhibit significant changes when malicious tampering occurs. To enable speech verification in a self-contained manner, the generated fingerprints are then embedded into the speech signal by segment-wise watermarking. Without a copy of the original authentic speech, SpeechVerifier can retrieve the fingerprint from the published audio and check it with the embedded watermark to verify the integrity of the speech. Our salient contributions are summarized as follows.

- We propose SpeechVerifier, a proactive speech verification design against tampering attacks, which enables users to verify speech integrity without accessing original speech recordings.
- To enable **self-contained verification**, we leverage audio watermarking to embed discriminative fingerprints into the speech signal, allowing for verifying the integrity only from the watermarked audio.
- We develop a five-step algorithm that extracts multiscale features and applies contrastive learning to generate binary fingerprints, which are **robust to benign** operations yet **sensitive to malicious** manipulations.
- Extensive experiments across diverse audio manipulation scenarios show that SpeechVerifier is effective in detecting tampering attacks and robust to benign operations.

2 Related works

Detect speech tampering passively based on acoustic features. Audio editing process can generate artifacts or modify natural acoustic features within human speech. For example, frame offset [33], inconsistent noise [18], and even discontinued electric network frequency [25, 6], are identified as evidence of tampering. Using such patterns, "passive" detectors can be trained as binary classifiers using labeled clean and tampered audio. However, these methods are less effective when facing deepfake audio. Advanced deepfake techniques can synthesize high-fidelity speech with few or no detectable artifacts, making conventional patterns unreliable. To address this, recent work has explored more subtle acoustic properties, such as fluid dynamics and articulatory phonetics [3]. Nevertheless, as the deepfake models evolve, relying solely on passive detection may not be sufficient against future attacks.

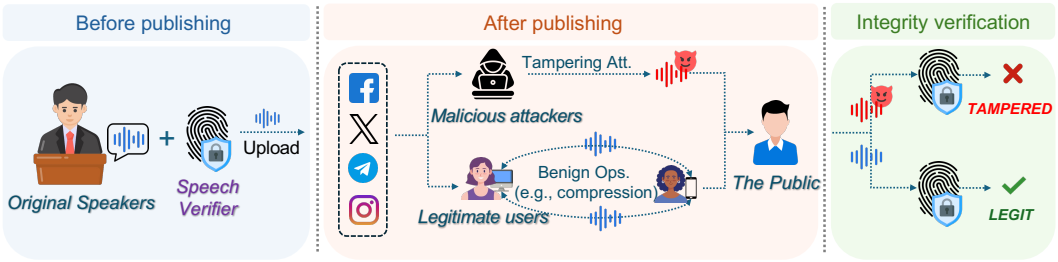


Figure 1: System overview of the proposed SpeechVerifier.

Protect genuine audio based on integrity verification. Proactive defense provides another direction to detect speech tampering. In general, critical information is extracted from the original audio and condensed into auxiliary data (or "meta" data). This auxiliary data then serves as a reference for verification: one extracts the same data from the test audio, and if it matches, it indicates that the test audio is free of audio editing, and vice versa. Cryptographic hashing, which transforms the digital audio files into discrete bytes, is one of the proactive defenses [34]. However, hashing operations are too sensitive to tolerate common operations from regular users, such as audio compression, resampling, resulting in a high false alarm rate. Another method is fragile watermarking [22], where sensitive watermarks are directly embedded into audio signals and checked for changes. However, this method is also sensitive to minor perturbations, limiting the free and practical distribution of audio. Ge et al. [8] propose a proactive defense approach against speaker identity manipulation, which embeds speaker embeddings into speech using audio watermarking. However, their method focuses only on speaker-identity attacks and cannot detect semantic content alterations. Therefore, existing proactive audio protection methods do not simultaneously achieve robustness against benign operations, sensitivity to malicious tampering, and independence from external verification channels.

3 Motivation

3.1 Problem Definition

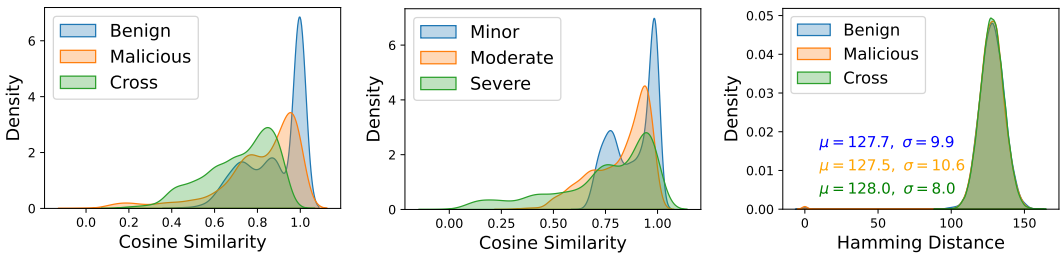
As shown in Figure 1, the scenario considered in our study includes four parties: 1) **Original speakers**, such as public institutions and celebrities, who publish statements or speeches on social media platforms. 2) **Legitimate users**, who download and repost these recordings to increase their dissemination. 3) **Malicious attackers**, which employ audio editing or voice conversion techniques, either changing the original semantic meaning or impersonating the speaker. 4) **The public**, who are exposed to conflicting audio sources, requiring a practical and reliable method to verify the integrity of a given speech recording.

3.2 Definition of Malicious Tampering vs. Benign Operations

We define tampering attacks as malicious audio modifications that alter the semantic content or the speaker identity. Typical malicious operations include audio splicing, deletion, substitution, silencing, text-to-speech (TTS) synthesis, and voice conversion. In contrast, benign operations refer to common audio transformations that occur during legitimate storage, transmission, or distribution processes, such as compression, reencoding, resampling, and noise suppression. These operations do not affect semantic content or speaker identity, and are therefore not considered tampering attacks. To better evaluate the impact of tampering, we further categorize malicious operations into three levels - Minor, Moderate, and Severe - based on their impacts on semantic content and speaker identity. A detailed distinction between malicious and benign audio operations, along with specific examples, is provided in Appendix A.2.

3.3 Limitations of Coarse-Grained Acoustic Feature Based Similarity Comparison

An intuitive approach to speech verification is to check whether the published speech audio sounds like the original one, which is technically to measure acoustic feature similarity. Following this intuition, we compared the similarity distributions of 3 categories: (i) the original audio and the audio modified by benign operations ("Benign"), (ii) the original audio and the audio modified by malicious operations ("Malicious"), and (iii) the original audio and the arbitrarily selected unrelated



(a) (b) (c)

Figure 2: Probability distributions: (a) wav2vec embedding similarity to the original audio under different modifications; (b) wav2vec embedding similarity to the original audio at different tampering levels; (c) SHA256 Hamming distance to the original audio under different modifications.

audio ("Cross"). Moreover, we compare the similarity distributions across malicious operations with varying degrees of tampering, as introduced earlier. Two types of acoustic features were used: Mel-frequency cepstral coefficients (MFCC) [5] and deep representations from wav2vec2.0 [1]. As shown in Figure 2a and Figure 2b, the similarity distributions based on wav2vec embeddings for "Benign" and "Malicious" overlap, and the similarity scores tend to decrease as the extent of tampering increases. This observation indicates that such similarity comparison can only measure the extent of modification, but cannot distinguish the types of modification operations (i.e., benign or malicious ones). Similar results using MFCC features are provided in Appendix B.3. These observations demonstrate that coarse-grained acoustic feature based similarity checking tends to ignore minor but semantically important speech edits. For instance, altering the phrase "do not" to "do" in a 20-second speech affects only about 0.2 seconds, and similarity remains more than 99%, while there is a huge semantic difference. Moreover, these methods require access to the original authentic audio, which is impractical in many scenarios. These observations highlight two key challenges:

Challenge 1: Insufficient sensitivity to semantic tampering attacks. The coarse-grained acoustic feature based similarity checking methods fail to distinguish benign operations from malicious ones, because they are not sensitive enough to semantic tampering attacks.

Challenge 2: Dependence on the original authentic audio. The similarity comparison methods have to use the original authentic audio as the reference, which is not always applicable in practice.

3.4 Limitations of Cryptographic Hashing Based Methods

Given Challenge 1, it is worth exploring some other speech verification methods that are sensitive to tampering attacks. A potential candidate is cryptographic hashing [14], which generates a digest of the entire audio file. Due to its high sensitivity, even the slightest modifications can be detected, so it can detect malicious tampering attacks effectively. However, such high sensitivity is not always positive in practice. Hash values may also change significantly under benign operations that do not affect either the semantic content or speaker identity. As shown in Figure 2c, the Hamming distances between original audio samples and their corresponding benign variants, malicious variants, and unrelated audio samples are all similarly large when using cryptographic hashing. This observation indicates that cryptographic hashing cannot distinguish benign operations from malicious tampering attacks. Moreover, hashing-based methods rely on external hashed results beyond the published audio itself for verification, which introduces extra cost. These observations expose two key challenges:

Challenge 3: Lack of robustness to benign operations. Hash values change significantly even under benign operations, making it too sensitive to use for speech verification.

Challenge 4: Dependence on external reference hash values. Hashing based methods rely on external reference hash values for speech verification. That introduces extra overhead and may not be convenient for speech forwarding on the online platforms in practice.

4 Methodology

4.1 SpeechVerifier Design Overview

To address those challenges, we propose SpeechVerifier, a proactive speech integrity verification design, which is (i) sensitive to tampering attacks, (ii) robust to benign operations, and (iii) convenient

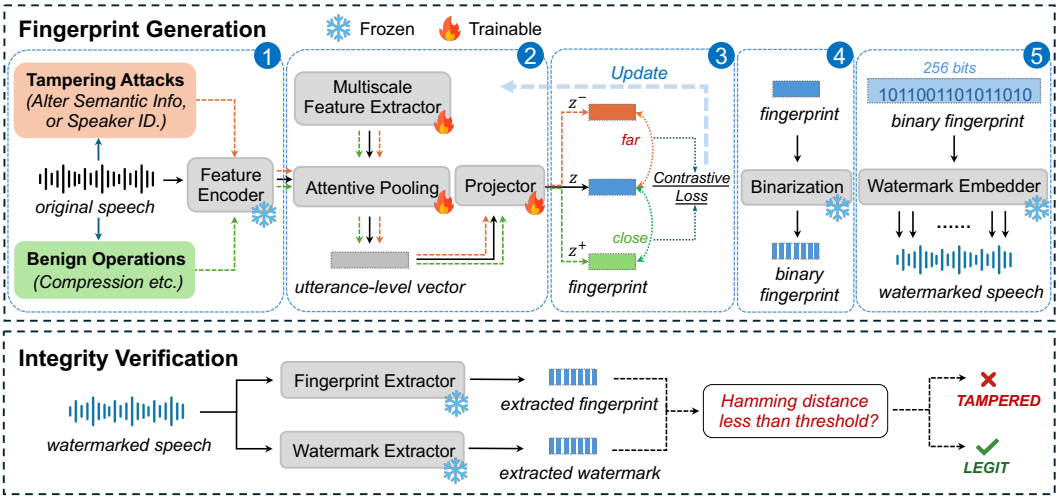


Figure 3: A sketch of the proposed SpeechVerifier design including speech fingerprint generation (top) and integrity verification (bottom).

to use by the public since it verifies the published speech audio's integrity in a self-contained manner. As the sketch shown in Figure 3, SpeechVerifier consists of two stages: fingerprint generation and dual-path integrity verification.

The speech fingerprint generation in SpeechVerifier has five steps: (1) **Frame-Level Feature Encoding (Speech to Representation)**: raw speech is encoded into frame-level representations that preserve acoustic information; (2) **Multiscale Acoustic Feature Extraction (Representation to Vector)**: the frame-level representations are first processed into contextual features, then aggregated at multiple temporal resolutions, and finally attentively pooled into a fixed-dimensional vector that summarizes the entire utterance; (3) **Contrastive Fingerprint Training (Vector to Fingerprint)**: the vector is optimized to be robust to benign operations, and sensitive to tampering attacks using contrastive learning; (4) **Binary Fingerprint Encoding (Fingerprint to Bit)**: the trained fingerprint is discretized into a binary representation; (5) **Segment-Wise Watermarking (Bit to Watermark)**: the binary fingerprint is embedded into the original audio through segment-wise watermarking, making the fingerprint self-contained.

The speech integrity verification in SpeechVerifier independently performs two parallel paths on the published audio: (1) regenerating the fingerprint via the same extraction pipeline, and (2) extracting the embedded watermark via the watermark decoding process. The two resulting binary codes are then compared using Hamming distance to determine whether the speech has been attacked.

4.2 Fingerprint Generation and Watermarking

Step 1. Frame-Level Feature Encoding (Speech to Representation) We utilize the pre-trained wav2vec 2.0 model [1] to extract frame-level representations from the original audio before publishing. This step serves as a necessary preprocessing stage for fingerprint generation. It converts continuous waveform signals into structured sequences of frame-level representations that preserve essential acoustic information. These representations have demonstrated effectiveness in downstream tasks such as automatic speech recognition [2] and speaker verification [7]. Formally, the feature encoder $\varepsilon : \mathcal{X} \to \mathcal{Z}$ maps raw audio waveforms $\mathcal{X}$ to a sequence of latent representations $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T$, where each $\mathbf{z}_t \in \mathbb{R}^{d_z}$ denotes the frame-level acoustic feature at time t, and T is the total number of output frames.

Step 2. Multiscale Acoustic Feature Extraction and Summarization (Representation to Vector). Given the frame-level representations $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T$ obtained from Step 1, this step constructs a fixed-dimensional vector that summarizes the speech across different temporal granularities. The multiscale feature extractor $\mathcal{F}$ consists of two components: (a) a bidirectional long short-term memory (BiLSTM) network that transforms the input frame-level representations into contextual hidden states, and (b) a multiscale pooling operation that averages the hidden states over phoneme-, word-, and

phrase-level windows (size 20, 50, and 100, respectively) , producing a sequence of multiscale features $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_K$ (see Appendix A.3).

To summarize these features into an utterance-level vector, we apply self-attentive pooling [13]. This mechanism assigns higher weights to more informative components, with attention weight computed as: $w_n = \frac{\exp(\phi(h_n))}{\sum_{t=1}^{K} \exp(\phi(h_t))}$, where $\phi(\cdot)$ is a feedforward network. The weighted sum yields a fixed-dimension vector: $\mathbf{v}' = \sum_{n=1}^{K} w_n \cdot h_n$, which is referred to as the utterance-level vector. To obtain a more compact representation for fingerprint optimization, a projection module is applied to reduce the dimensionality of $\mathbf{v}'$, yielding the final fingerprint vector $\mathbf{v} \in \mathbb{R}^{d_v}$.

Step 3. Contrastive Fingerprint Training (Vector to Fingerprint). Given the fixed-length vector $\mathbf{v}$ obtained from Step 2, we optimize it to serve as a distinctive audio fingerprint that is robust to benign operations and sensitive to malicious tampering attacks. To this end, we adopt contrastive learning [17] to guide the training of all preceding modules. During the training, a batch of original speech samples is randomly selected, where each sample serves as an anchor. For each anchor, we generate: **Positive pairs**, consisting of the anchor and its benign variants (e.g., compression and re-encoding), and **Negative pairs**, consisting of the anchor and its tampered variants (e.g., substitution and deletion). Detailed operations are listed in Appendix A.2. The contrastive loss is defined as

$$\mathcal{L}_c = -\frac{1}{B}\sum_{i=1}^{B}\frac{1}{P}\sum_{j=1}^{P}\log\frac{\exp\left(\tilde{\mathbf{v}}_i^{\text{Orig.}\top}\tilde{\mathbf{v}}_{i,j}^{\text{Benign}}/\tau\right)}{\sum_{k=1,\ k\neq i}^{N}\exp\left(\tilde{\mathbf{v}}_i^{\text{Orig.}\top}\tilde{\mathbf{v}}_{i,k}/\tau\right)}, \tag{1}$$

where B is the number of anchors in the batch, P is the number of benign variants per anchor, and N denotes the total number of comparison samples for each anchor, including its own benign and tampered variants as well as embeddings from other anchors in the batch. τ is the temperature parameter. $\tilde{\mathbf{v}}_i^{\text{Orig.}}$ denotes the L2-normalized embedding of the i-th anchor, $\tilde{\mathbf{v}}_{i,j}^{\text{Benign}}$ denotes the embedding of its j-th benign variant, and $\tilde{\mathbf{v}}_{i,k}$ enumerates all embeddings in the batch, including benign, tampered and unrelated samples.

This contrastive learning above encourages the model to bring the anchor closer to its benign variants while pushing it away from tampered and unrelated samples in the embedding space. As a result, the fixed-length vector is optimized to serve as a distinctive audio fingerprint that is robust to benign operations while remaining sensitive to malicious tampering attacks.

Step 4. Binary Fingerprint Encoding (Fingerprint to Bit). Compared to full-precision vectors in continuous space, binary representations are more suitable for compact storage (e.g., embedding into a watermark) and fast retrieval (e.g., through bit-wise comparison). Therefore, we convert the continuous fingerprint vector $\mathbf{v} \in \mathbb{R}^{d_v}$ into a binary code $\mathbf{b} \in \{-1, +1\}^d$. This binary fingerprint can directly be embedded into the audio signal. Specifically, we apply a `tanh` activation followed by a `sign` function at the final projection layer to obtain the binarized output. As demonstrated later in the evaluation, the binarized fingerprint preserves its discriminative characteristics of $\mathbf{v}$, i.e., robust to benign operations while sensitive to tampering attacks.

Step 5. Segment-Wise Watermarking (Bit to Watermark). To enable self-contained verification, we embed the binary fingerprint directly into the speech signal. Inspired by AudioSeal [26], we aim to develop a high-capacity and inaudible audio watermarking method based on the Encodec framework. While AudioSeal targets short watermarks for copyright-protection (i.e., 16 bits), SpeechVerifier must embed much longer fingerprints (e.g., 256 bits) to support speech integrity verification. To meet this requirement, we propose a segment-wise watermarking scheme. Given an input waveform $\mathcal{X}$ of duration T seconds and its binary fingerprint $\mathbf{b}$, both are divided into N non-overlapping segments:

$$\mathcal{X} = [\mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(N)}] \quad \text{and} \quad \mathbf{b} = [\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(N)}], \tag{2}$$

where each $\mathcal{X}^{(n)}$ spans T/N seconds and each $\mathbf{b}^{(n)}$ contains d/N bits. For each audio segment $\mathcal{X}^{(n)}$, we embed $\mathbf{b}^{(n)}$ into the Encodec embedding space and generate a watermark signal $\delta^{(n)}$. The watermarked segment is then formed as: $\tilde{\mathcal{X}}^{(n)} = \mathcal{X}^{(n)} + \delta^{(n)}$. Finally, the watermarked segments $[\tilde{\mathcal{X}}^{(1)}, \ldots, \tilde{\mathcal{X}}^{(N)}]$ are concatenated, yielding the final self-verifiable audio. Notably, since watermark only incurs subtle perturbations and remains imperceptible to human listeners, the embedded binary fingerprint can be reliably extracted from the watermarked speech audio $\tilde{\mathcal{X}}$ without degradation.

4.3 Dual-Path Speech Integrity Verification

SpeechVerifier employs a dual-path mechanism to assess the integrity of the published speech $\tilde{\mathcal{X}}$:

Path A: Fingerprint Generation from Published Speech. The published speech audio is processed using the same fingerprint generation pipeline described in Section 4.2. The fingerprint $\mathbf{b}'$ is computed as $\mathbf{b}' = \text{sign}(\mathcal{F}(\varepsilon(\tilde{\mathcal{X}})))$, where ε and $\mathcal{F}$ denote the feature encoder and multiscale extractor, respectively, and $\text{sign}(\cdot)$ denotes the final binarization function.

Path B: Watermark Extraction. The published speech audio $\tilde{\mathcal{X}}$ is processed in the inverse manner of Step 5 to decode the embedded watermark (i.e., the original binary fingerprint). From each segment $\tilde{\mathcal{X}}^{(n)}$, we extract the bit chunk $\hat{\mathbf{b}}^{(n)}$ using the watermark decoder, and then reconstruct the full watermark as $\hat{\mathbf{b}} = [\hat{\mathbf{b}}^{(1)}, \hat{\mathbf{b}}^{(2)}, \ldots, \hat{\mathbf{b}}^{(N)}]$.

Finally, the integrity of the published audio is verified by comparing the generated fingerprint $\mathbf{b}'$ with the extracted watermark $\hat{\mathbf{b}}$. This is done by computing the Hamming distance as follows.

$$d_H(\mathbf{b}', \hat{\mathbf{b}}) \leq \theta \quad \Rightarrow \quad \text{Accept}; \quad \text{otherwise} \quad \text{Reject},$$

where θ is a decision threshold set based on the development data from public datasets.

5 Experiments

5.1 Experiment Setup

Dataset. To train and evaluate the performance of the proposed SpeechVerifier, we use **VoxCeleb1** [15] dataset, which includes over 150,000 utterances from 1,251 celebrities. These audio samples are collected from interviews and public videos, providing the conditions that reflect real-world speech recordings. In addition, we use the test subset from **LibriSpeech** [19] dataset. LibriSpeech is a corpus of approximately 1,000 hours of English read speech, sourced from public domain audiobooks. This setup allows us to perform a cross-domain evaluation to assess model generalization. More details about the datasets and the preprocessing steps are provided in Appendix B.2.

Implementation details. Fingerprint model: We use the pre-trained Wav2Vec2.0 Base model as the acoustic feature extractor, obtained from the official repository[2]. A two-layer BiLSTM with a hidden size of 512 follows the feature extractor. Multiscale pooling is used with window sizes of 20, 50, and 100 frames with a stride of 10 frames. A two-layer projection head then maps features into a 256-dimensional vector. **Watermark model:** The pre-trained AudioSeal model[3] is used to embed and extract fingerprints as watermark payloads. To improve the watermarking capacity, we divide both the carrier audio and the fingerprint into 16 segments. Each segment carries a 16-bit watermark, leading to a total payload of 256 bits per audio sample. **Train:** We exploit benign and malicious operations (see Appendix A.2) and the original audio samples for contrastive learning, with temperature set as 0.05. A cosine annealing learning rate schedule is used, gradually decreasing the learning rate from 1×10^{-3} to 1×10^{-5} over the training.

Evaluation Metrics. The threshold in Section 4.3 is determined on the development dataset ($\theta = 42$). We compare the dual-path bit error against θ for binary classification, where benign and malicious operations are treated as positive and negative samples, respectively. Evaluation metrics include true positive rate (TPR), false positive rate (FPR), true negative rate (TNR), false negative rate (FNR), equal error rate (EER), and area under the curve (AUC). Additionally, we present data visualizations, as well as cosine similarity and Hamming distance analyses, to demonstrate effectiveness.

5.2 Results

Robustness to benign operations. Table 1 presents the performance of the proposed SpeechVerifier in accepting published speech samples subjected to benign audio operations, as defined in Appendix A.2. We focus on evaluating how well SpeechVerifier accepts positive samples with harmless modifications (TPR) and whether it mistakenly accepts maliciously tampered speech (FPR). The numbers of positive and negative test samples are balanced in Table 1. When trained and evaluated on different subsets of VoxCeleb, SpeechVerifier achieves 100% TPR and 0% FPR across *all* listed benign operations,

[2]https://github.com/facebookresearch/fairseq/tree/main/examples/wav2vec

[3]https://github.com/facebookresearch/audioseal

Table 1: Results of benign operation (positive) acceptance on VoxCeleb and LibriSpeech.

Operation	VoxCeleb				LibriSpeech				Semantic	Identity
	TPR	FPR	AUC	EER	TPR	FPR	AUC	EER		
Compression	1.00	0.00	1.00	0.00	1.00	0.03	1.00	0.01	✓	✓
Reencoding	1.00	0.00	1.00	0.00	1.00	0.02	1.00	0.01	✓	✓
Resampling	1.00	0.00	1.00	0.00	0.97	0.01	1.00	0.02	✓	✓
Noise suppression	1.00	0.00	1.00	0.00	0.98	0.01	1.00	0.01	✓	✓
Overall	1.00	0.00	1.00	0.00	0.99	0.02	1.00	0.02	-	-

Table 2: Results of malicious operation (negative) rejection on VoxCeleb and LibriSpeech.

Operation	VoxCeleb				LibriSpeech				Semantic	Identity
	TNR	FNR	AUC	EER	TNR	FNR	AUC	EER		
Deletion	1.00	0.00	1.00	0.00	0.99	0.00	1.00	0.00	✗	✓
Splicing	1.00	0.00	1.00	0.00	0.99	0.08	0.99	0.01	✗	✓
Silencing	1.00	0.00	1.00	0.00	0.98	0.00	1.00	0.01	✗	✓
Substitution	1.00	0.00	1.00	0.00	0.99	0.00	1.00	0.01	✗	✓
Reordering	1.00	0.00	1.00	0.00	0.98	0.00	1.00	0.00	✗	✓
Text-to-speech	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	✗	✓
Voice conversion	1.00	0.00	1.00	0.00	0.98	0.00	1.00	0.01	✓	✗
Overall	1.00	0.00	1.00	0.00	0.98	0.01	1.00	0.02	-	-

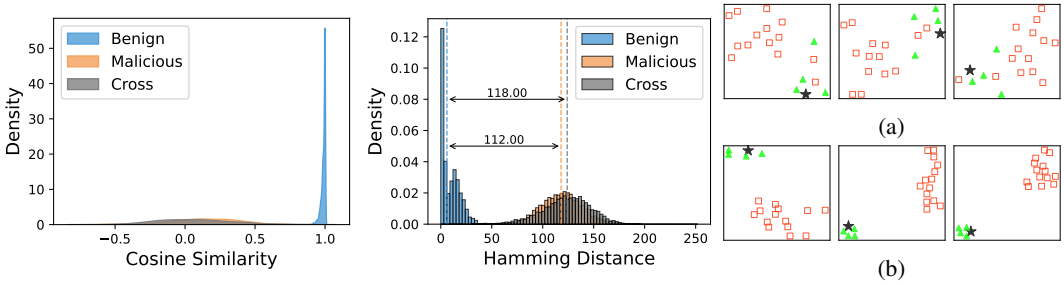
demonstrating strong robustness to non-malicious transformations. For cross-dataset evaluation on LibriSpeech, using a model trained on VoxCeleb, the overall TPR/FPR slightly change to 99% and 2%, respectively, indicating good generalizability across datasets.

Sensitivity to tampering attacks. Table 2 evaluates the ability of **SpeechVerifier** to reject malicious tampering attacks that alter the semantic content or speaker identity. Specifically, we consider tampering attacks including deletion, splicing, silencing, substitution, reordering, as well as deepfake-based manipulations such as text-to-speech synthesis (TTS) and voice conversion, as detailed in Appendix A.2. In this evaluation, tampering operations (actual negatives) to the published audio are expected to be rejected with a high true negative rate, while minimizing the false negative rate, which reflects incorrect rejection of benign samples. Notably, SpeechVerifier achieves 100% and 98% overall TNR, and 0% and 1% overall FNR on the VoxCeleb (in-domain) and LibriSpeech (cross-domain) datasets, respectively. These results highlight SpeechVerifier's strong sensitivity to tampering attacks. A more detailed breakdown by tampering strength (e.g., minor, moderate and severe) is provided in Appendix B.5.

5.3 Evaluation of Multiscale Feature and Binarized Fingerprints

Figure 4 presents two types of analysis: (a) cosine similarity between extracted multiscale features and (b) Hamming distance between binarized fingerprints. In Figure 4a, benign-original pairs show high similarity values close to 1.0, while malicious-original and cross-original pairs exhibit significantly lower similarity scores, indicating that the learned multiscale features effectively capture the differences between benign operations and malicious tampering attacks. Here, "cross" refers to the arbitrarily selected unrelated audio samples. In Figure 4b, binarized fingerprints of benign processed samples yield low Hamming distances from their corresponding retrieved watermarks, whereas malicious and cross pairs have much larger distances, with a clear separation gap of nearly 112-118 bits. This indicates that the binarization process preserves discriminability and can differentiate tampering attacks from benign operations by using a simple threshold.

Figure 5 illustrates the t-SNE visualizations of the extracted multiscale features before and after training. Specifically, Figure 5a and Figure 5b show the relative distribution of anchors (original speech), positives (after benign operations), and negatives (after malicious operations) in the latent space. Before training, anchor and positive samples are scattered and overlap with negatives, indicating that the initial features extracted are not well separated. After training, anchors and positives form tight clusters, while negatives are clearly separated. This suggests that contrastive learning-trained multiscale feature extraction effectively learns embeddings that distinguish benign operations from malicious tampering, which explains the strong performance of SpeechVerifier. Additional visualization evidences are provided in Appendix B.8.



(a) (b)

Figure 4: (a) Extracted feature similarity; (b) binarized fingerprint Hamming distance.

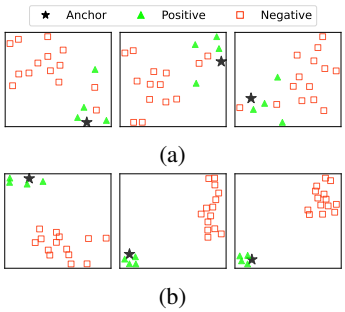


Figure 5: t-SNE visualizations of speech samples: (a) before training; (b) after training.

5.4 Comparison with Other Deepfake Detection Methods

We finally evaluate SpeechVerifier as a deepfake detector[4]. Specifically, we compare SpeechVerifier with state-of-the-art methods including RawNet2 [29] and AASIST [10], two end-to-end models developed for the ASVspoof challenge [32], and widely used for audio spoofing detection. We utilize a zero-shot TTS model YourTTS [4] to generate deepfake speech segments, and substitute them for varying proportions (10%, 25%, 50%, 75% and 90%) of the original speech. Next, the deepfake samples are mixed with an equal amount of clean speech to ensure fair evaluation.

From Table 3, both RawNet2 and AASIST perform best given the highest substitution ratio at 90%, achieving up to 83% TPR by RawNet2. However, when decreasing the substitution ratio, RawNet2 and AASIST both show significant degradation regarding the ability of detecting deepfake substitution samples. For example, at the ratio of 10%, the AUC of RawNet2 drops to 65% and EER increases to 38%, indicating diminished sensitivity to subtle spoofing. Similarly, AASIST exhibits similar performance degradation under the same condition. In contrast, SpeechVerifier constantly achieves very good detection performance across all substitution levels (TPR=1.00, FPR=0.00, AUC=1.00, EER=0.00), demonstrating the superiority of the proposed proactive defense design.

Since synthetic deepfake audio lacks embedded watermarks, the fingerprint-watermark verification process becomes essentially random, making tampering attacks easy to detect. Even minor substitutions alter the extracted fingerprint and disrupt the embedded watermark simultaneously, resulting in a mismatch and enabling reliable detection of tampering. Further analysis of each module's contributions is provided in the ablation studies in Appendix B.7.

Table 3: Performance of Deepfake detection at varying substitution ratios

Deepfake Ratio	**RawNet2**				**AASIST**				**SpeechVerifier (ours)**			
	TPR	**FPR**	**AUC**	**EER**	**TPR**	**FPR**	**AUC**	**EER**	**TPR**	**FPR**	**AUC**	**EER**
Substitute 10%	0.58	0.34	0.65	0.38	0.39	0.17	0.64	0.39	**1.00**	**0.00**	**1.00**	**0.00**
Substitute 25%	0.62	0.34	0.68	0.37	0.37	0.17	0.64	0.42	**1.00**	**0.00**	**1.00**	**0.00**
Substitute 50%	0.59	0.34	0.65	0.38	0.44	0.17	0.67	0.39	**1.00**	**0.00**	**1.00**	**0.00**
Substitute 75%	0.68	0.34	0.72	0.34	0.54	0.17	0.75	0.31	**1.00**	**0.00**	**1.00**	**0.00**
Substitute 90%	0.83	0.34	0.81	0.27	0.62	0.17	0.76	0.33	**1.00**	**0.00**	**1.00**	**0.00**

6 Conclusion

In this paper, we have proposed SpeechVerifier, a proactive speech integrity verification design. SpeechVerifier employs multiscale feature extraction and contrastive learning to generate robust acoustic fingerprints. Then, it embeds the generated fingerprints into audio signals through watermarking. The fingerprint is designed to be sensitive to malicious tampering attacks but robust to benign operations, thus not affecting the normal audio distribution and usage. This approach enables the general public to conveniently verify the published speech audio's integrity in a self-contained manner, i.e., by extracting and comparing embedded watermark messages with corresponding acoustic fingerprints. Extensive experiments demonstrate that SpeechVerifier achieves very good detection performance under various tampering attack scenarios while staying robust to benign operations.

[4]To avoid confusion, SpeechVerifier is used here for deepfake detection, where "positive" now refers to deepfake samples to be identified.

A SpeechVerifier Design and Operation Definitions

A.1 Overall Algorithm

Algorithm 1 SpeechVerifier Training and Deployment

Input: Raw speech $\mathcal{X}$, benign operations $\mathcal{T}_b(\cdot)$, malicious operations $\mathcal{T}_m(\cdot)$, Wav2Vec2.0 encoder ε, multiscale feature extractor $\mathcal{F}$
Output: Watermarked speech $\tilde{\mathcal{X}}$
for $e = 1, 2, \ldots,$ epochs **do**
  for $b = 1, 2, \ldots,$ batches **do**
    $\mathcal{X}^{\text{benign}} \leftarrow \mathcal{T}_b(\mathcal{X}), \quad \mathcal{X}^{\text{malicious}} \leftarrow \mathcal{T}_m(\mathcal{X})$ ▷ Generate positive and negative variants
    Step 1: Frame-level feature extraction
    $\mathcal{Z} \leftarrow \varepsilon(\mathcal{X})$ ▷ Extract frame-level acoustic representations
    Step 2: Multiscale feature summarization
    $\mathbf{h}_n \leftarrow \mathcal{F}(\mathcal{Z})$ ▷ BiLSTM and multiscale pooling
    for $n = 1, \ldots, K$ **do**
      $w_n \leftarrow \frac{\exp(\phi(\mathbf{h}_n))}{\sum_{t=1}^{K} \exp(\phi(\mathbf{h}_t))}$ ▷ Attentive weight computation
    end for
    $\mathbf{v}' \leftarrow \sum_{n=1}^{K} w_n \cdot \mathbf{h}_n$ ▷ Utterance-level representation
    $\mathbf{v} \leftarrow \text{Proj}(\mathbf{v}')$ ▷ Final fingerprint vector
    Step 3: Contrastive fingerprint training
    Compute contrastive loss $\mathcal{L}_c$
    Update $\mathcal{F}$, ϕ, Proj via backpropagation
  end for
end for
Step 4: Binary fingerprint encoding
$\mathbf{b} \leftarrow \text{sign}(\tanh(\text{Proj}(\text{AttPool}(\mathcal{F}(\varepsilon(\tilde{X}))))))$ ▷ Extract binary fingerprint
Step 5: Segment-wise watermarking
Split $\mathcal{X}$ and $\mathbf{b}$ into N segments: $[\mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(N)}]$, $[\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(N)}]$
for $n = 1, \ldots, N$ **do**
  $\delta^{(n)} \leftarrow \text{WatermarkEmbedder}(\mathcal{X}^{(n)}, \mathbf{b}^{(n)})$
  $\tilde{\mathcal{X}}^{(n)} \leftarrow \mathcal{X}^{(n)} + \delta^{(n)}$ ▷ Embed watermark into audio
end for
$\tilde{\mathcal{X}} \leftarrow \text{Concat}(\tilde{\mathcal{X}^{(1)}}, \ldots, \tilde{\mathcal{X}}^{(N)})$ ▷ Get watermarked audio

Algorithm 2 SpeechVerifier Verification

Input: Published speech $\tilde{\mathcal{X}}$, wav2vec2.0 encoder ε, trained multiscale feature extractor $\mathcal{F}$, projection module Proj, attentive pooling AttPool, WatermarkExtractor
Output: Verification result (Accept or Reject)
Path A: Fingerprint extraction
$\mathbf{b}' \leftarrow \text{sign}(\tanh(\text{Proj}(\text{AttPool}(\mathcal{F}(\varepsilon(\tilde{\mathcal{X}}))))))$
Path B: Segment-wise watermark extraction
Split $\tilde{\mathcal{X}}$ into N segments: $\tilde{\mathcal{X}}^{(1)}, \ldots, \tilde{\mathcal{X}}^{(N)}$
for $n = 1$ to N **do**
  $\hat{\mathbf{b}}^{(n)} \leftarrow \text{WatermarkExtractor}(\tilde{\mathcal{X}}^{(n)})$
end for
$\hat{\mathbf{b}} \leftarrow \text{Concat}(\hat{\mathbf{b}}^{(1)}, \ldots, \hat{\mathbf{b}}^{(N)})$
Integrity decision
if $d_H(\mathbf{b}', \hat{\mathbf{b}}) \leq \theta$ **then**
  return Accept
else
  return Reject
end if

A.2 Definition of Benign Operations and Malicious Tampering

We simulate two categories of audio modifications: *benign operations* and *malicious tampering*. Benign operations refer to legitimate processing steps encountered during audio storage, transmission, or distribution. These operations do not change the semantic content or the speaker identity of the speech. In contrast, malicious tampering refers to intentional alterations designed to distort either the semantic meaning or the identity of the speaker. We detail each operation below and summarize their characteristics in Table 4.

Table 4: Summary of audio operations.

Operation	Example	Implementation
Benign Operations		
Compression	Podcasts, news broadcasts, online meetings	`ffmpeg -i in.wav -b:a 128k tmp.mp3; ffmpeg -i tmp.mp3 out.wav`
Reencoding	Saving or uploading audio files	`ffmpeg -i in.wav out.wav`
Resampling	Low-bandwidth communication	`torchaudio.transforms.Resample`
Noise Suppression	Social media platforms	`RMS-based frame muting`
Malicious Operations		
Deletion	Removing "not" in "I do not agree"	`VAD + remove voiced portion`
Splicing	Inserting "not" into "I do agree"	`Insert voiced segment`
Substitution	Replacing "agree" with "disagree"	`Swap waveform segment`
Silencing	Muting "not" in "I do not agree"	`Mute VAD-detected region`
Reordering	Changing sentence order	`Segment + shuffle + concat`
Voice Conversion	Changing timbre (speaker identity)	`torchaudio.sox_effects`
Text-to-Speech	Generate new speech with speaker's timbre	`YourTTS (zero-shot synthesis)`

Compression. Lossy compression is applied by converting the waveform to MP3 at 128 kbps and decoding it back to WAV. This simulates typical processing in podcasts and streaming platforms. We use FFmpeg: `ffmpeg -i input.wav -b:a 128k temp.mp3; ffmpeg -i temp.mp3 output.wav`.

Reencoding. The waveform is re-encoded to 16-bit PCM WAV format without compression. This simulates storage or uploading scenarios where minor numerical alterations may occur. Implemented with: `ffmpeg -i input.wav output.wav`.

Resampling. Audio is downsampled (e.g., from 16 kHz to 8 kHz) and then upsampled back, simulating low-bandwidth or legacy systems. Implemented with: `torchaudio.transforms.Resample`.

Noise Suppression. To simulate automatic noise suppression utilized by social media and streaming platforms, the waveform is divided into overlapping frames. Frames with low root-mean-square (RMS) energy are muted.

Deletion. A portion of speech (not silence) is removed from the speech. For example, deleting "not" from "I do not agree" changes the meaning entirely.

Splicing. A short segment of speech from the same speaker is spliced into the waveform. For example, inserting "not" into the phrase "I do agree" reverses its original semantic meaning.

Substitution. A segment of speech is replaced with another waveform segment of equal length from the same speaker. For instance, replacing "agree" with "disagree" fundamentally changes the intended meaning.

Silencing. A portion of speech (not silence or noise) is deliberately muted by setting its amplitude to zero. For instance, muting the word "not" in "I do not agree" leads to a reversed interpretation.

Reordering. The speech is segmented, rearranged, and concatenated to change the semantic content. For instance, reordering "I never said she stole my money" into "She stole my money, I never said" distorts the original meaning and can lead to an opposite interpretation.

Voice Conversion. The speech is manipulated to alter its speaker identity while preserving the linguistic content. We simulate this by shifting pitch (e.g., +4 semitones) using SoX effects. This modification can make the utterance sound like it was spoken by someone else. We implement this using `torchaudio.sox_effects.apply_effects_tensor`.

Text-to-Speech. We synthesize speech from text using a pre-trained text-to-speech (TTS) model, YourTTS[5] [4], which supports multilingual and zero-shot speaker adaptation. This attack can generate speech that closely mimics the speaker's voice with arbitrary semantic content.

Different Levels of Tampering. To evaluate the performance under varying conditions, we define three levels of tampering: *Minor*, *Moderate*, and *Severe*. Specifically, at the Minor level, tampering operations — including deletion, splicing, silencing, and substitution — alter about 10% of the original audio content (alteration ratio = 0.1). At the Moderate level, these same operations alter 30% of the audio (alteration ratio = 0.3). At the Severe level, 50% of the audio is altered (alteration ratio = 0.5), and this level also includes reordering operations, which disrupts the logical structure of the speech.

A.3 Explanation of Malicious Tampering over Different Granularities

Table 5 presents representative examples of malicious tampering at the phoneme, word, and phrase levels. These examples illustrate how manipulations at different temporal granularities can alter the meaning of speech. They also motivate the use of multiscale pooling with window sizes of 20, 50, and 100 frames, which are designed to capture such variations in real-world scenarios.

Table 5: Examples of malicious tampering at different levels of granularity

Granularity	Example	Description
Phoneme-level	Change "bed" to "bad" (English); change "mā" (mother) to "mǎ" (horse) (Mandarin)	Altering a single phoneme or syllable can lead to subtle yet meaningful changes. These edits are often difficult to detect but can reverse or distort the intended meaning.
Word-level	Insert "not" into "He is guilty" to form "He is not guilty"; replace "approved" with "denied"	Tampering at the word level through insertion, deletion, or substitution can directly modify semantic content, leading to misleading interpretations.
Phrase-level	Change "Negotiations will begin immediately" to "Negotiations will be delayed indefinitely"	Reordering or replacing entire phrases can fabricate new narratives while maintaining natural-sounding speech, making the tampering more deceptive.

[5] `https://github.com/Edresson/YourTTS`

B Experimental Setup and Extended Results

B.1 Implementation Details

To supplement Section 5.1, we provide a detailed description of the model architecture and training configuration.

Model. We use the pretrained Wav2Vec2.0 Base model [6] to extract 768-dimensional frame-level acoustic features. These are passed to a two-layer Bidirectional LSTM (BiLSTM) with an input size of 768, a hidden size of 512 (i.e., 256 per direction), and a dropout rate of 0.25. To capture temporal features at multiple resolutions, we apply average pooling with window sizes of 20, 50, and 100 frames, with a stride of 10 frames, implemented using `avg_pool1d` along the time axis. The pooled outputs are aggregated by an attentive pooling module consisting of a linear-tanh-linear projection. The resulting weighted sum forms the utterance-level embedding, followed by dropout with a rate of 0.2. This embedding is fed into a two-layer MLP projection head with dimensions $768 \to 512 \to 256$, with ReLU activation between layers. The final output vector is L2-normalized and passed through a $\tanh$ function to constrain values to the range $[-1, 1]$, yielding the continuous-valued fingerprint. For segment-wise watermarking, we use the pretrained AudioSeal model[7] to embed and extract binary fingerprints as watermarks. Each audio is divided into 16 non-overlapping segments, with each segment embedded with a 16-bit binary watermark, resulting in a total payload size of 256 bits per audio.

Training. SpeechVerifier is trained using a cosine annealing learning rate schedule, decaying from 1×10^{-3} to 1×10^{-5} over 50 epochs. The contrastive loss is temperature-scaled with $\tau = 0.05$. Training is conducted on 4 NVIDIA Quadro RTX 8000 GPUs using distributed data parallelism (DDP).

B.2 Dataset and Evaluation Details

We use two public speech datasets: **VoxCeleb** and **LibriSpeech**. For VoxCeleb, the development set is used for training and the test set for evaluation. For LibriSpeech, we use only the `test-clean` subset for evaluation. All audio files are converted to WAV format and resampled to 16 kHz.

Preprocessing. We randomly sample 5,000 utterances from the VoxCeleb development set for model training. For evaluation, we sample 500 utterances each from the VoxCeleb test set and the LibriSpeech `test-clean` subset. To stabilize the training and ensure data quality, we retain only utterances with durations between 2 and 20 seconds.

For each valid utterance, we generate two sets of augmented variants for contrastive learning:

Benign Augmentations: These are modifications that preserve both speaker identity and semantic content. The details can be found in Appendix A.2.

Malicious Augmentations: These include tampering operations intended to alter speaker identity and semantic content. The details can be found in Appendix A.2.

Evaluation Metric For evaluation, we consider benign and malicious as positive and negative classes, respectively. TP is the number of benign samples correctly classified, and FN is the number of benign samples incorrectly classified as malicious. FP is the number of malicious samples incorrectly classified as benign, and TN is the number of malicious samples correctly rejected. The following metrics are computed:

- True Positive Rate (TPR):

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

- False Positive Rate (FPR):

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}},$$

[6] `https://github.com/facebookresearch/fairseq/blob/main/examples/wav2vec`

[7] `https://github.com/facebookresearch/audioseal`

- True Negative Rate (TNR):

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}},$$

- False Negative Rate (FNR):

$$\text{FNR} = \frac{\text{FN}}{\text{FN} + \text{TP}}$$

- Equal Error Rate (EER): The error rate at the decision threshold where FPR = FNR.
- Area Under the ROC Curve (AUC): Computed by integrating the Receiver Operating Characteristic (ROC) curve over various thresholds.

B.3 Similarity Distribution using MFCC Feature

To complement the observations in Section 3.3, we present similarity distributions computed using handcrafted Mel-frequency cepstral coefficients (MFCC) instead of wav2vec2 embeddings. As shown in Figure 6, the similarity distributions between original and modified audio samples using MFCC features exhibit trends similar to those observed with wav2vec2-based representations. Specifically, the distributions corresponding to benign and malicious modifications overlap, and the similarity scores tend to decrease as the extent of tampering increases. This indicates that MFCC-based similarity comparison can only measure the extent of modification but does not effectively distinguish between different types of modifications.

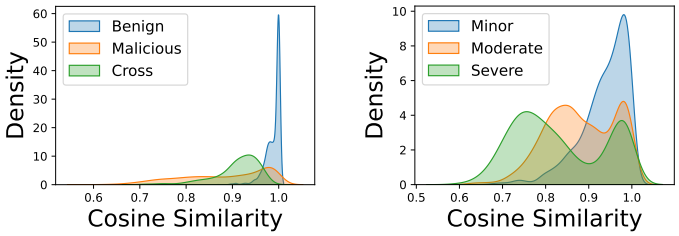


Figure 6: Probability distributions: (a) MFCC embedding similarity to the original audio under different modifications; (b) MFCC embedding similarity to the original audio at different tampering levels.

B.4 Evaluation on Semantic and Identity Changes under Benign and Malicious Operations

We evaluate the impact of different audio modifications on both semantic integrity and speaker identity consistency. Semantic preservation is quantified using word error rate (WER) computed from a pre-trained automatic speech recognition (ASR) model[8], `facebook/wav2vec2-base-960h`, a CTC-based ASR model. Speaker identity preservation is measured by cosine similarity between embeddings extracted using the pre-trained speaker verification (SV) model[9], `speechbrain/spkrec-ecapa-voxceleb`.

As shown in Table 6, benign operations (e.g., compression, recording, resampling, noise suppression) result in low WER ($\leq$8.24) and high identity similarity ($\geq$0.78), indicating that they largely preserve both semantic content and speaker identity.

In contrast, malicious operations introduce substantial degradation. WER increases steadily with the severity of deletion, splicing, silencing, and substitution, reflecting significant semantic changes. These operations, however, generally maintain high identity similarity because they retain the original timbre. Notably, voice conversion results in relatively low WER, but significantly reduces identity similarity (41.60%), since it deliberately alters the speaker's timbre.

To further investigate the nonzero WER observed under benign operations, we manually examined the ASR outputs. Most transcription errors were minor substitutions or alignment shifts that did not affect the overall meaning. This suggests that the observed WER in these cases reflects limitations of the ASR model and metric sensitivity rather than genuine semantic distortion.

Table 6: WER and Identity Similarity under Different Operations

Operation	**WER %**	**Identity Similarity %**
Benign Operations		
Compression	1.15	95.60
Recoding	0.26	99.99
Resampling	6.87	78.00
Noise suppression	8.24	94.52
Malicious Operations		
Deletion (minor)	21.65	99.04
Deletion (moderate)	40.20	96.84
Deletion (severe)	62.32	93.29
Splicing (minor)	31.24	99.00
Splicing (moderate)	52.45	97.35
Splicing (severe)	78.36	96.55
Silencing (minor)	30.76	98.16
Silencing (moderate)	53.79	90.13
Silencing (severe)	75.74	75.96
Substitution (minor)	23.22	98.33
Substitution (moderate)	48.12	94.36
Substitution (severe)	63.03	90.72
Reordering	69.55	99.53
Text-to-speech	-	-
Voice conversion	8.60	**41.60**

[8] `https://github.com/facebookresearch/fairseq/blob/main/examples/wav2vec`

[9] `https://huggingface.co/speechbrain/spkrec-ecapa-voxceleb`

B.5 Results of Fine-Grained Malicious Operations Rejection

Table 7: Results of fine-grained malicious operation rejection on VoxCeleb and LibriSpeech.

Operation	VoxCeleb				LibriSpeech				Semantic	Identity
	TNR	FNR	AUC	EER	TNR	FNR	AUC	EER		
Deletion (minor)	1.00	0.00	1.00	0.00	0.98	0.00	1.00	0.00	✗	✓
Deletion (moderate)	1.00	0.00	1.00	0.00	0.99	0.00	1.00	0.00	✗	✓
Deletion (severe)	1.00	0.00	1.00	0.00	0.99	0.00	1.00	0.00	✗	✓
Splicing (minor)	1.00	0.00	1.00	0.00	0.99	0.16	0.98	0.06	✗	✓
Splicing (moderate)	1.00	0.00	1.00	0.00	0.99	0.07	0.99	0.03	✗	✓
Splicing (severe)	1.00	0.01	1.00	0.01	0.99	0.01	1.00	0.01	✗	✓
Silencing (minor)	1.00	0.00	1.00	0.00	0.99	0.00	1.00	0.00	✗	✓
Silencing (moderate)	1.00	0.00	1.00	0.00	0.98	0.00	1.00	0.00	✗	✓
Silencing (severe)	1.00	0.00	1.00	0.00	0.98	0.00	1.00	0.02	✗	✓
Substitution (minor)	1.00	0.00	1.00	0.00	0.99	0.00	1.00	0.01	✗	✓
Substitution (moderate)	1.00	0.00	1.00	0.00	0.99	0.00	1.00	0.01	✗	✓
Substitution (severe)	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	✗	✓
Reordering	1.00	0.00	1.00	0.00	0.98	0.00	1.00	0.00	✗	✓
Text-to-speech	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	✗	✓
Voice conversion	1.00	0.00	1.00	0.00	0.98	0.00	1.00	0.01	✓	✗
Overall	1.00	0.00	1.00	0.00	0.99	0.02	1.00	0.01	-	-

B.6 Watermarked Speech Quality

Table 8: Audio quality metrics.

Methods	SI-SNR	PESQ	STOI	LSD
SpeechVerifier	25.14	4.28	0.998	0.111

We evaluate the perceptual quality of watermarked speech using four objective metrics. (1)Scale-Invariant Signal to Noise Ratio (SI-SNR) quantifies waveform-level distortion in decibels (dB). Higher values indicate less distortion. (2)Perceptual Evaluation of Speech Quality (PESQ) [24] ranges from 1.0 (poor) to 4.5 (excellent), and reflects perceived speech quality. (3)Short-Time Objective Intelligibility (STOI) [28] ranges from 0 to 1, with higher values indicating better intelligibility. (4)Log Spectral Distance (LSD) measures spectral deviation between original and watermarked speech, lower values indicate greater spectral fidelity.

As shown in Table 8, our proposed SpeechVerifier has little perceptual degradation. The high SI-SNR and PESQ scores, along with near-perfect intelligibility (STOI) and low spectral error (LSD), demonstrate that the watermarking process preserves both fidelity and intelligibility, making it suitable for practical deployment.

B.7 Ablation Studies

SpeechVerifier consists of four core modules: an acoustic feature encoder, a multiscale feature extractor, an attentive pooling module, and a contrastive learning objective. To assess the contribution of each module, we conduct ablation studies by replacing each module with alternatives and evaluating the performance.

For the feature encoder, we compare handcrafted Mel-Frequency Cepstral Coefficients (MFCC) with deep speech representations obtained from a pretrained wav2vec model. To evaluate the impact of the multiscale feature extractor, we remove the multiscale pooling, directly use the output of BiLSTM. For temporal pooling, we substitute attentive pooling with average pooling. Regarding the loss function, we compare InfoNCE with Triplet Loss, which are widely used in contrastive learning.

As shown in Table 9, each module contributes to the overall performance. Replacing the wav2vec encoder with handcrafted MFCC features results in only a slight performance drop, suggesting that MFCC can also serve as a lightweight alternative. Removing the multiscale feature extractor leads to a significant degradation, indicating the importance of extracting both global and local temporal

patterns. Substituting attentive pooling with average pooling reduces performance, indicating that the attention mechanism provides better frame selection for embedding generation. Finally, replacing InfoNCE with Triplet Loss results in a substantial performance decline, demonstrating that InfoNCE is more effective for learning discriminative embeddings in our task.

Table 9: Ablation studies on acoustic feature encoder, feature extractor, temporal pooling scheme, and loss function.

Method Variant	**TPR**	**FPR**	**AUC**	**EER**
SpeechVerifier (wav2vec → MFCC)	0.9979	0.0021	0.9999	0.0021
SpeechVerifier (Multiscale → w/o Multiscale)	0.7443	0.1224	0.8847	0.1958
SpeechVerifier (AttentivePooling → AvgPooling)	0.9875	0.0163	0.9988	0.0144
SpeechVerifier (InfoNCEloss → TripletLoss)	0.8594	0.0146	0.9577	0.1073
SpeechVerifier	**0.9979**	**0.0016**	**1.0000**	**0.0013**

B.8 Visualization of Multiscale Feature Extraction

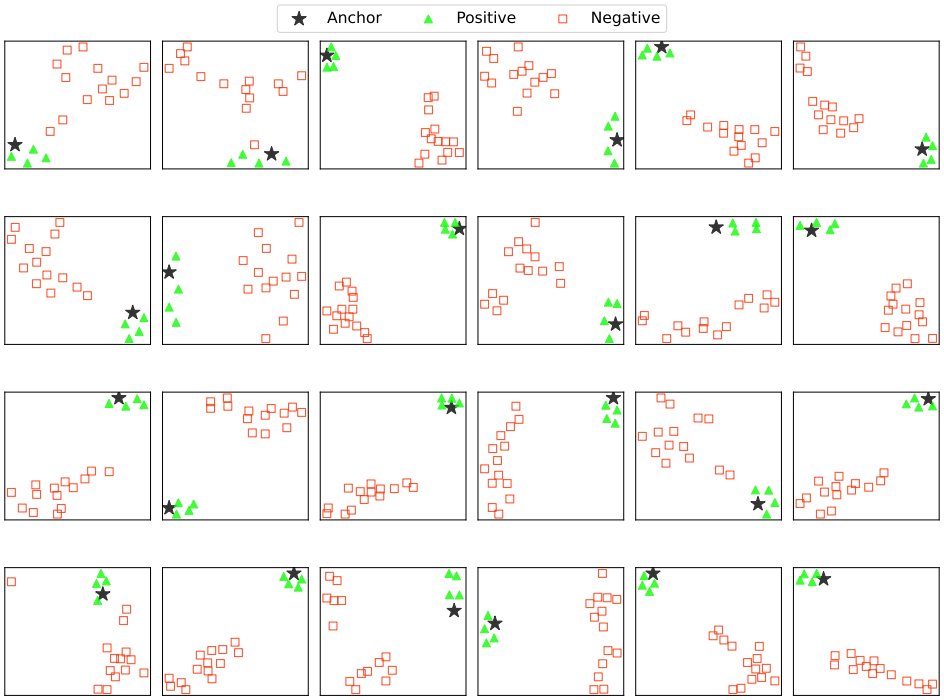


Figure 7: t-SNE visualizations of speech samples (after training).

C Limitations and Broader Impact

C.1 Limitations

While SpeechVerifier demonstrates strong robustness to benign audio operations and effective detection of malicious tampering, there are several aspects that can be further improved:

- SpeechVerifier currently identifies whether an audio has been tampered with, but does not localize the exact region of tampering. Future work could explore providing localized detection for more precise forensic analysis.
- Although we have performed cross-dataset evaluation using LibriSpeech, our experiments mainly focus on English speech under relatively clean conditions. The generalization of SpeechVerifier to noisy, multilingual, and more diverse real-world tampering operations needs further investigation.
- SpeechVerifier assumes that the input audio is sufficiently long ($\geq$ 2s) to support segment-wise watermark embedding and extraction. For very short audio clips, the process of embedding and extracting watermarks may become unreliable. Moreover, such short clips often lack meaningful semantic content and offer limited value for tampering, making protection less critical in practice. However, to extend the applicability of SpeechVerifier, future work is needed to explore more reliable verification methods for short speech segments.

C.2 Broader Impact

SpeechVerifier aims to proactively protect the integrity of publicly shared speech content. By detecting audio tampering and impersonation attacks, it can effectively mitigate the spread of misinformation, safeguard individual reputations, and uphold public trust in digital media. Furthermore, since SpeechVerifier operates without relying on external references or original recordings, it substantially reduces verification costs and enhances scalability.

References

[1] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020.

[2] Alexei Baevski, Wei-Ning Hsu, Alexis Conneau, and Michael Auli. Unsupervised speech recognition. *Advances in Neural Information Processing Systems*, 34:27826–27839, 2021.

[3] Logan Blue, Kevin Warren, Hadi Abdullah, Cassidy Gibson, Luis Vargas, Jessica O'Dell, Kevin Butler, and Patrick Traynor. Who are you (i really wanna know)? detecting audio {DeepFakes} through vocal tract reconstruction. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 2691–2708, 2022.

[4] Edresson Casanova, Julian Weber, Christopher D Shulby, Arnaldo Candido Junior, Eren Gölge, and Moacir A Ponti. Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone. In *International Conference on Machine Learning*, pages 2709–2720. PMLR, 2022.

[5] Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980.

[6] Paulo Antonio Andrade Esquef, José Antonio Apolinário, and Luiz WP Biscainho. Edit detection in speech recordings via instantaneous electric network frequency variations. *IEEE Transactions on Information Forensics and Security*, 9(12):2314–2326, 2014.

[7] Zhiyun Fan, Meng Li, Shiyu Zhou, and Bo Xu. Exploring wav2vec 2.0 on speaker verification and language identification. In *Proc. Interspeech 2021*, pages 1509–1513, 2021.

[8] Wanying Ge, Xin Wang, and Junichi Yamagishi. Proactive detection of speaker identity manipulation with neural watermarking. In *The 1st Workshop on GenAI Watermarking*, 2025.

[9] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. In *International Conference on Machine Learning*, pages 13916–13932. PMLR, 2023.

[10] Jee-weon Jung, Hee-Soo Heo, Hemlata Tak, Hye-jin Shim, Joon Son Chung, Bong-Jin Lee, Ha-Jin Yu, and Nicholas Evans. Aasist: Audio anti-spoofing using integrated spectro-temporal graph attention networks. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6367–6371. IEEE, 2022.

[11] Awais Khan, Khalid Mahmood Malik, James Ryan, and Mikul Saravanan. Voice spoofing countermeasures: Taxonomy, state-of-the-art, experimental analysis of generalizability, open challenges, and the way forward. *arXiv preprint arXiv:2210.00417*, 2022.

[12] Daniele Ugo Leonzio, Luca Cuccovillo, Paolo Bestagini, Marco Marcon, Patrick Aichroth, and Stefano Tubaro. Audio splicing detection and localization based on acquisition device traces. *IEEE Transactions on Information Forensics and Security*, 18:4157–4172, 2023.

[13] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[14] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 2018.

[15] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. Voxceleb: A large-scale speaker identification dataset. *Interspeech 2017*, 2017.

[16] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[17] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[18] Xunyu Pan, Xing Zhang, and Siwei Lyu. Detecting splicing in digital audios using local noise level estimation. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1841–1844. IEEE, 2012.

[19] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[20] Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller. Deep voice 3: Scaling text-to-speech with convolutional sequence learning. In *International Conference on Learning Representations*, 2018.

[21] The Washington Post. Ai is spawning a flood of fake trump and harris voices. `https://www.washingtonpost.com/technology/interactive/2024/ai-voice-detection-trump-harris-deepfake-election/`, 2024. Accessed: 2025-04-22.

[22] Diego Renza, Camilo Lemus, et al. Authenticity verification of audio signals based on fragile watermarking for audio forensics. *Expert systems with applications*, 91:211–222, 2018.

[23] Reuters. Fact check: Video does not show joe biden making transphobic remarks. `https://www.reuters.com/article/fact-check/video-does-not-show-joe-biden-making-transphobic-remarks-idUSL1N34Q1IW/`, 2023. Accessed: 2025-04-22.

[24] Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*, volume 2, pages 749–752. IEEE, 2001.

[25] Daniel Patricio Nicolalde Rodríguez, José Antonio Apolinario, and Luiz Wagner Pereira Biscainho. Audio authenticity: Detecting enf discontinuity with high precision phase analysis. *IEEE Transactions on Information Forensics and Security*, 5(3):534–543, 2010.

[26] Robin San Roman, Pierre Fernandez, Hady Elsahar, Alexandre Défossez, Teddy Furon, and Tuan Tran. Proactive detection of voice cloning with localized watermarking. In *Proceedings of the 41st International Conference on Machine Learning*, pages 43180–43196, 2024.

[27] Martin Steinebach and Jana Dittmann. Watermarking-based digital audio data authentication. *EURASIP Journal on Advances in Signal Processing*, 2003:1–15, 2003.

[28] Cees H Taal, Richard C Hendriks, Richard Heusdens, and Jesper Jensen. A short-time objective intelligibility measure for time-frequency weighted noisy speech. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4214–4217. IEEE, 2010.

[29] Hemlata Tak, Jose Patino, Massimiliano Todisco, Andreas Nautsch, Nicholas Evans, and Anthony Larcher. End-to-end anti-spoofing with rawnet2. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6369–6373. IEEE, 2021.

[30] Yuancheng Wang, Zeqian Ju, Xu Tan, Lei He, Zhizheng Wu, Jiang Bian, et al. Audit: Audio editing by following instructions with latent diffusion models. *Advances in Neural Information Processing Systems*, 36:71340–71357, 2023.

[31] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *Interspeech 2017*, page 4006, 2017.

[32] Junichi Yamagishi, Xin Wang, Massimiliano Todisco, Md Sahidullah, Jose Patino, Andreas Nautsch, Xuechen Liu, Kong Aik Lee, Tomi Kinnunen, Nicholas Evans, et al. Asvspoof 2021: accelerating progress in spoofed and deepfake speech detection. *arXiv preprint arXiv:2109.00537*, 2021.

[33] Rui Yang, Zhenhua Qu, and Jiwu Huang. Detecting digital audio forgeries by checking frame offsets. In *Proceedings of the 10th ACM Workshop on Multimedia and Security*, pages 21–26, 2008.

[34] Mohammed Zakariah, Muhammad Khurram Khan, and Hafiz Malik. Digital multimedia audio forensics: past, present and future. *Multimedia Tools and Applications*, 77:1009–1040, 2018.