SPEECHECK: SELF-CONTAINED SPEECH INTEGRITY VERIFICATION VIA EMBEDDED ACOUSTIC FINGERPRINTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Advances in audio editing have made public speeches increasingly vulnerable to malicious tampering, raising concerns for social trust. Existing speech tampering detection methods remain insufficient: they often rely on external references or fail to balance sensitivity to attacks with robustness against benign operations like compression. To tackle these challenges, we propose SpeeCheck, the first self-contained speech integrity verification framework. SpeeCheck can (i) effectively detect tampering attacks, (ii) remain robust under benign operations, and (iii) enable direct verification without external references. Our approach begins with utilizing 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 self-contained verification, these fingerprints are embedded into the audio itself via a watermark. Finally, during verification, SpeeCheck retrieves the fingerprint from the audio and checks it with the embedded watermark to assess integrity. Extensive experiments demonstrate that SpeeCheck reliably detects tampering while maintaining robustness against common benign operations. Real-world evaluations further confirm its effectiveness in verifying speech integrity. The code and demo are available at https://speecheck.github.io/SpeeCheck/.

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 (Wang et al., 2023) and text-to-speech (TTS) generation models (Wang et al., 2017; Ping et al., 2018; Huang et al., 2023; Du et al., 2024; Chen et al., 2024) 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 (Reuters, 2023; Post, 2024). 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 authenticity from numerous sources. 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 can be categorized into two groups: passive detection and proactive protection. Passive detection methods (Rodríguez et al., 2010; Yang et al., 2008; Pan et al., 2012; Blue et al., 2022; Leonzio et al., 2023) rely on deep binary classifiers trained to identify artifacts introduced by tampering. 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,

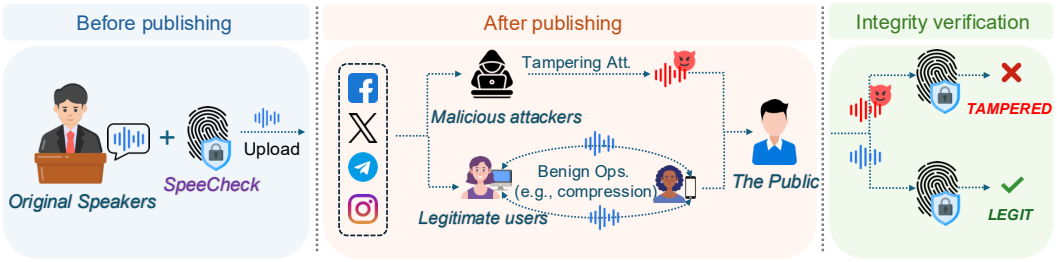
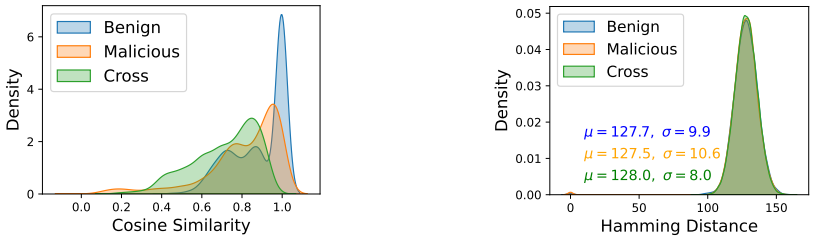


Figure 1: System overview of the proposed SpeeCheck.

leaving systems vulnerable to impersonation-based attacks (Khan et al., 2022). Proactive protection methods ensure content integrity by embedding auxiliary information into the audio and extracting it during verification. Common approaches include cryptographic hashing (Steinebach & Dittmann, 2003) and fragile watermarking (Renza et al., 2018). Hash-based methods convert audio into fixed-length digests, which can reliably detect even minor alterations. However, they require transmitting or retrieving external reference hashes, preventing independent verification from the published audio. Fragile watermarking, on the other hand, embeds a highly sensitive watermark into the audio, enabling integrity verification without references. However, they can be unintentionally removed by benign operations, which restricts applicability in real-world distribution scenarios. By contrast, robust watermarking (Roman et al., 2024; Liu et al., 2024; Chen et al., 2023) is designed to survive such benign processing and is widely used for copyright protection. Yet, it is not suitable for integrity verification, as the watermark may remain detectable after malicious tampering.

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 in Instagram/TikTok), ensuring usability in sharing and distribution. Therefore, in this paper, we propose SpeeCheck, a proactive acoustic fingerprint-based speech verification design that jointly utilizes semantic content and speaker identity. Specifically, SpeeCheck 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, SpeeCheck can retrieve the fingerprint from the published audio and check it with the embedded watermark to verify the integrity. Our main contributions are summarized as follows.

- We propose SpeeCheck, the first self-contained integrity verification framework for speech. It 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.
- We validate SpeeCheck through extensive experiments on public speech datasets and a real-world dataset constructed for this study. The evaluation demonstrates high effectiveness in detecting diverse tampering attacks while maintaining robustness against benign operations in practical scenarios.



(a) (b)

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

2 Motivation

2.1 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 help disseminate these audio recordings by downloading or reposting them. 3) **Malicious attackers**, which employ audio editing or voice conversion techniques to alter either the semantic content or the speaker's identity. 4) **The public**, who are exposed to conflicting audio sources, requires a reliable method to verify the integrity of a given speech recording.

2.2 Malicious and Benign Audio Operations

We define malicious audio tampering as intentional audio modifications that alter the semantic content or 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 processes such as storage, transmission, or distribution. Examples include compression, reencoding, resampling, and noise suppression, none of which impact the semantic content or speaker identity. A detailed distinction between malicious and benign audio operations, along with specific examples, is provided in Appendix B.2.

2.3 Limitations of Acoustic Feature Similarity

An intuitive approach for speech verification is to compare the acoustic similarity between the published audio and its original version. Following this intuition, we analyzed similarity scores between the original audio and three types of modifications: benignly processed variants ("Benign"), maliciously modified variants ("Malicious"), and unrelated audio samples ("Cross"). Figure 2a presents cosine similarity distributions computed using wav2vec embeddings (Baevski et al., 2020). The significant overlap between benign and malicious similarity distributions demonstrates that acoustic feature similarity alone is insufficient to determine the types of modification operations. Similar results are observed using traditional acoustic feature Mel-frequency cepstral coefficients (MFCC) (Davis & Mermelstein, 1980), detailed in Appendix C.3. This limitation arises because malicious operations, even significantly altering the content, may introduce minimal acoustic changes. For instance, modifying the phrase "do not" to "do" in a 20-second speech affects only 0.2 seconds, and similarity remains more than 99%, while causing substantial semantic alteration. Moreover, this method requires access to the authentic audio, which is impractical in real-world scenarios. These limitations highlight two key challenges:

Challenge 1: Insufficient sensitivity to semantic tampering attacks. Acoustic feature-based similarity 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. Acoustic similarity assessments require the original authentic audio as a reference, which is not always applicable in practice.

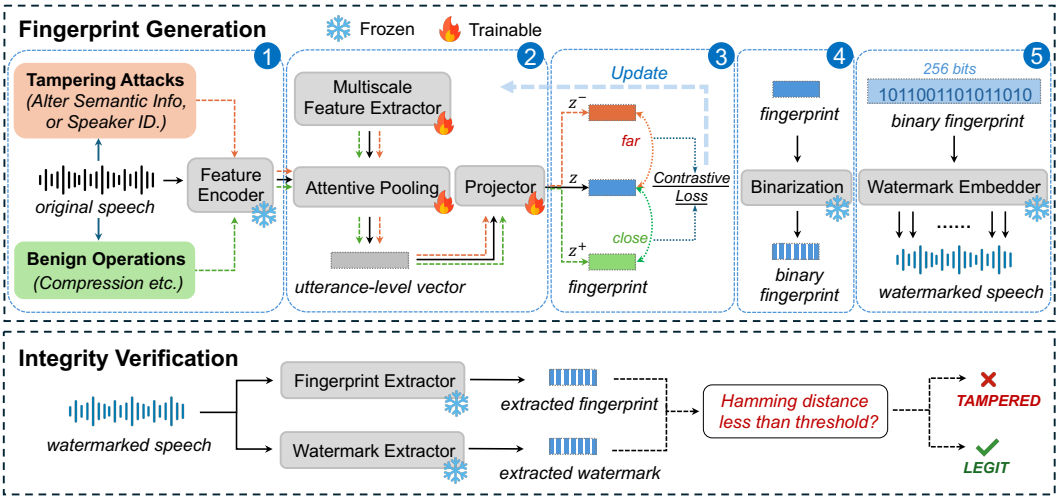


Figure 3: A sketch of the proposed SpeeCheck design including speech fingerprint generation (top) and integrity verification (bottom).

2.4 Limitations of Cryptographic Hashing

Given these limitations, another potential verification method is cryptographic hashing (Menezes et al., 2018), which generates a digest for each audio file. Its extreme sensitivity enables the detection of even minor changes, making it theoretically effective against tampering. However, this sensitivity also captures benign operations that do not affect semantic content or speaker identity. Figure 2b illustrates the significant Hamming distances between original audio and benign, malicious, or unrelated variants. Consequently, cryptographic hashing fails to differentiate benign operations from malicious tampering. Moreover, cryptographic hashing requires external reference hash values for verification, introducing additional practical complexity. These limitations expose two key challenges:

Challenge 3: Lack of robustness to benign operations. Cryptographic hashes are overly sensitive, changing significantly even under benign operations, thus limiting practical usability.

Challenge 4: Dependence on external reference hash values. Verification using hashes depends on externally stored hash values, introducing extra overhead and inconvenience for speech forwarding on the online platforms.

3 Methodology

3.1 SpeeCheck Overview

To address these challenges, we propose SpeeCheck, a proactive speech integrity verification design, which is (i) sensitive to tampering attacks, (ii) robust to benign operations, and (iii) convenient 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, SpeeCheck consists of two stages: fingerprint generation and dual-path integrity verification.

The speech fingerprint generation in SpeeCheck 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 integrity verification in SpeeCheck 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 decoder. The two resulting binary codes are then compared using Hamming distance to determine whether the speech has been attacked.

3.2 Fingerprint Generation and Watermarking

Step 1. Frame-Level Feature Encoding (Speech to Representation) We utilize the pre-trained wav2vec 2.0 model (Baevski et al., 2020) 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 (Baevski et al., 2021) and speaker verification (Fan et al., 2021). 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 (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 B.3 for examples).

To summarize these features into an utterance-level vector, we apply self-attentive pooling (Lin et al., 2017). 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 (Oord et al., 2018) 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 negative pairs, consisting of the anchor and its tampered variants (e.g., substitution). Detailed operations are listed in Appendix B.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). To enable self-contained verification, we embed the generated fingerprint into the audio signal as a watermark. Since watermarking schemes typically support binary payloads and inevitably incur information loss, we design the fingerprint representation to preserve its discriminative power even after binarization. Specifically, we convert the continuous fingerprint vector $\mathbf{v} \in \mathbb{R}^{d_v}$ into a binary code $\mathbf{b} \in \{-1, +1\}^d$, which is more suitable for compact storage and fast retrieval via bit-wise comparison. To encourage the output to approach the bipolar extremes of -1 and +1 and thus reduce quantization error, we apply a `tanh` activation at the final projection layer. This is followed by a `sign` function to obtain the final binarized output. As demonstrated in Section 4.2, the binarized fingerprint retains 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, the binary fingerprint must be embedded directly into the speech signal. We adapt the robust watermarking method AudioSeal (Roman et al., 2024) for this purpose. However, a key challenge arises from our high-capacity requirement. While AudioSeal is designed for short watermarks (i.e., 16 bits) for copyright protection, our task requires embedding much longer fingerprints (e.g., 256 bits). To meet this requirement, we extend the original AudioSeal with a segment-wise embedding strategy. 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, the watermark incurs only subtle perturbations. Our experiments confirm that the acoustic fingerprint generated from the watermarked audio $\tilde{\mathcal{X}}$ remains consistent with the original fingerprint, while the embedded bits can still be reliably extracted without degradation.

3.3 Dual-Path Speech Integrity Verification

SpeeCheck 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 before. 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 validation set from public datasets.

4 Experiments

4.1 Experiment Setup

Dataset. To train and evaluate the performance of SpeeCheck, we use VoxCeleb1 (Nagrani et al., 2017), which includes over 150,000 utterances from 1,251 celebrities. These audio samples are collected from interviews and public videos, providing conditions that reflect real-world speech recordings. We further employ the test subset from LibriSpeech (Panayotov et al., 2015) dataset to assess the model generalization. Furthermore, to validate SpeeCheck's effectiveness under authentic

scenarios, we build a real-world speech dataset and evaluated it after fine-grained editing and distribution across major social media platforms. More details about the datasets and the preprocessing steps are provided in Appendix C.2.

Implementation details. (i) Fingerprint model: We use Wav2Vec2.0 Base model[1] as the acoustic feature extractor. 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. (ii) Watermark model: AudioSeal model[2] 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. (iii) Training: We exploit benign and malicious operations (see Appendix B.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. We evaluate SpeeCheck as a binary classification task, where benign operations are treated as positive and malicious ones as negative. 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) (see Appendix C.2). The decision is made by comparing the dual-path bit error with a threshold ($\theta = 42$), which is determined on the validation set.

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	99.80	1.60	99.84	1.40	97.00	2.40	99.51	2.80	✓	✓
Reencoding	99.60	1.20	99.86	0.60	98.60	3.40	99.80	2.20	✓	✓
Resampling	95.80	0.00	99.96	1.30	94.60	2.20	99.35	3.20	✓	✓
Noise suppression	99.60	0.80	99.95	0.30	98.40	1.80	99.85	1.70	✓	✓
Overall	98.70	0.90	99.90	0.90	97.15	2.45	99.63	2.48	-	-

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	99.47	0.00	100.00	0.07	98.80	0.00	100.00	0.13	✗	✓
Splicing	99.67	0.00	100.00	0.00	98.63	0.00	100.00	0.20	✗	✓
Silencing	99.23	0.47	99.84	0.87	97.57	1.73	99.63	2.57	✗	✓
Substitution	97.83	4.33	99.70	2.20	94.63	7.27	98.90	4.10	✗	✓
Reordering	98.40	2.40	99.08	2.60	97.20	3.00	99.05	3.00	✗	✓
Text-to-speech	100.00	0.00	100.00	0.00	100.00	0.00	100.00	0.00	✗	✓
Voice conversion	99.40	0.00	100.00	0.00	97.80	0.00	100.00	0.00	✓	✗
Overall	99.14	1.03	99.80	0.82	97.80	1.71	99.65	1.43	-	-

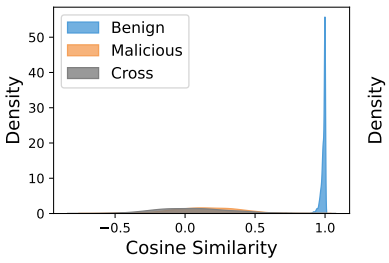
4.2 RESULTS

Robustness to benign operations. Table 1 presents the performance of the proposed SpeeCheck in accepting published speech samples subjected to benign audio operations, as defined in Appendix B.2. We focus on evaluating how well SpeeCheck accepts positive samples with harmless modifications (TPR) and whether it mistakenly accepts maliciously tampered speech (FPR). To ensure balanced evaluation, the numbers of positive (benign) and negative (malicious) samples are kept equal. On the test subsets of VoxCeleb1, SpeeCheck achieves an overall TPR of 98.70% and an FPR of 0.90%, demonstrating strong robustness to non-malicious transformations. For cross-dataset evaluation on LibriSpeech, using a model trained on VoxCeleb1, the TPR/FPR slightly change to 97.15% and 2.45%, respectively, indicating good generalizability across datasets.

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

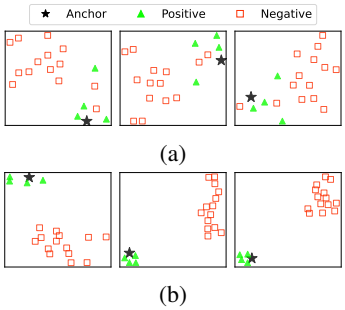
[2] https://github.com/facebookresearch/audioseal

Sensitivity to tampering attacks. Table 2 evaluates the ability of SpeeCheck to reject malicious tampering attacks that alter the semantic content or speaker identity. The considered attacks include simple audio editing (e.g., deletion) as well as advanced learning-based manipulations such as text-to-speech (TTS) synthesis and voice conversion, as detailed in Appendix B.2. In this setting, tampering operations (actual negatives) are expected to be rejected with a high true negative rate (TNR), while minimizing the false negative rate (FNR), which reflects incorrect rejection of benign samples. Notably, on the VoxCeleb (in-domain) dataset, SpeeCheck achieves an overall TNR of 99.14% and an FNR of 1.03%. On the LibriSpeech dataset, the system maintains strong performance with an overall of 97.80% and an FNR of 1.71%. These results highlight SpeeCheck's strong sensitivity to tampering attacks. A more detailed breakdown by tampering strength (e.g., minor, moderate and severe) is provided in Appendix C.5.



(a) (b)

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



(a) (b)

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

Multiscale feature and binarized fingerprints analysis. Figure 4 shows two analyses: (a) cosine similarity between extracted multiscale features and (b) Hamming distance between binarized fingerprints. In Figure 4a, benign-original pairs yield high similarity values close to 1.0, while malicious-original and cross-original pairs are much lower, 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 retrieved watermarks, whereas malicious and cross pairs have much larger distances, with a clear margin of about 112-118 bits. This indicates that the binarization process preserves discriminability and enables reliable separation of tampering and benign operations using a simple threshold.

Figure 5 shows the t-SNE visualizations of the extracted multiscale features before and after training. Specifically, Figure 5a and Figure 5b show the 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 poor separability. After training, anchors and positives form tight clusters, while negatives are clearly separated. This suggests that contrastive learning enables the multiscale feature extractor to learn embeddings that distinguish benign operations from malicious tampering, which explains the strong performance of SpeeCheck. Additional visualization evidences are provided in Appendix C.11.

Table 3: Detection accuracy on unseen benign operations and tampering attacks.

Unseen Operation Type	**Accuracy (%)**	d_{wm}	d_{fp}
Benign			
Loudness Normalization	100.00	0.96	0.08
Room Reverb	100.00	3.12	26.60
Combined Benign	100.00	4.64	8.29
Tampering			
Voice Changer (Female)	100.00	122.46	71.62
Voice Changer (Male)	100.00	125.38	67.85
Combined Malicious	100.00	119.32	75.08

Table 4: Performance of Deepfake detection at varying substitution ratios (all values in percentage)

Deepfake Ratio	RawNet2				AASIST				SpeeCheck (ours)			
	TPR	FPR	AUC	EER	TPR	FPR	AUC	EER	TPR	FPR	AUC	EER
Substitute 10%	58.12	34.07	65.43	38.29	39.21	17.36	64.58	39.14	**100.00**	**0.00**	**100.00**	**0.00**
Substitute 25%	61.87	33.95	68.24	37.41	37.46	17.18	64.12	41.89	**100.00**	**0.00**	**100.00**	**0.00**
Substitute 50%	59.38	34.26	65.72	38.11	44.19	17.09	67.34	39.27	**100.00**	**0.00**	**100.00**	**0.00**
Substitute 75%	68.54	34.11	72.36	34.28	54.07	16.92	75.48	31.24	**100.00**	**0.00**	**100.00**	**0.00**
Substitute 90%	82.91	34.33	81.27	27.18	62.38	17.24	76.19	33.42	**100.00**	**0.00**	**100.00**	**0.00**

Out-of-Distribution and real-world generalization. To evaluate SpeeCheck's generalization to unseen audio operations, we conduct out-of-distribution (OOD) experiments using operations not included in the training set. Table 3 reports detection accuracy at a fixed threshold ($\theta = 42$), along with the Hamming distances of watermarks (d_{wm}) and fingerprints (d_{fp}) between modified and original audio. SpeeCheck achieves 100% accuracy on unseen benign transformations, including loudness normalization, room reverberation, and combined benign manipulations. Concurrently, it successfully detects sophisticated unseen attacks, including commercial voice changer tools (ElevenLabs, 2024) that alter speaker identity and combined malicious edits. These results highlight the effectiveness of SpeeCheck in identifying unseen audio operations. Specifically, for real-world recordings, we observe similar strong performance, where SpeeCheck reliably identifies all fine-grained tamperings. Detailed evaluation is shown in Appendix C.6.

Deepfake detection comparison. We finally evaluate SpeeCheck as a deepfake detector[3]. Specifically, we compare SpeeCheck with state-of-the-art methods including RawNet2 (Tak et al., 2021) and AASIST (Jung et al., 2022), two end-to-end models developed for the ASVspoof challenge (Yamagishi et al., 2021), and widely used for audio spoofing detection. We utilize a zero-shot TTS model YourTTS (Casanova et al., 2022) 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 4, both RawNet2 and AASIST perform best given the highest substitution ratio at 90%, achieving up to 82.91% 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 65.43% and EER increases to 38.29%, indicating diminished sensitivity to subtle spoofing. Similarly, AASIST exhibits similar performance degradation under the same condition. In contrast, SpeeCheck consistently achieves very good detection performance across all substitution levels (TPR=100.00, FPR=0.00, AUC=100.00, 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 C.10.

5 Conclusion

In this paper, we proposed **SpeeCheck**, a proactive and self-contained framework for speech integrity verification. SpeeCheck integrates multiscale feature extraction and contrastive learning to produce robust fingerprints, which are embedded into audio via watermarking. These fingerprints are sensitive to malicious tampering while robust to benign operations commonly introduced during digital distribution, enabling integrity verification without access to external references. Extensive experiments confirm its robustness and sensitivity across diverse tampering scenarios. Notably, evaluations on a constructed real-world dataset further demonstrate its practicality, showing high robustness under social media distribution and strong sensitivity to fine-grained malicious edits.

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

Ethics Statement

This work does not involve human subjects, personally identifiable information, or sensitive data. All experiments are conducted on publicly available datasets (VoxCeleb and LibriSpeech) and a small-scale real-world dataset collected with voluntary consent. To protect privacy, all data used in public demos are anonymized, and no personally identifiable information is released. Deepfake and voice conversion technologies are employed solely to simulate attack scenarios for research evaluation, and no harmful or deceptive content is created or disseminated.

The proposed method aims to strengthen speech integrity verification and mitigate the spread of misinformation. We recognize that, like any integrity verification technology, it could be misused for surveillance or censorship; thus, it should be deployed responsibly and transparently. The authors declare no conflicts of interest or sponsorship-related concerns in this study.

Reproducibility Statement

We make significant efforts to ensure reproducibility. All datasets used in this study are publicly available (VoxCeleb, LibriSpeech), and the constructed real-world dataset is included in the supplementary materials. Details of the fingerprint generation, watermark embedding, training procedure, and evaluation metrics are described in Section 3 and Section 4, with extended information in the Appendix C. An anonymous implementation and demo are provided at `https://speecheck.github.io/SpeeCheck/`, which contains the source code and instructions for reproducing our experiments.

References

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.

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

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)*, pp. 2691–2708, 2022.

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*, pp. 2709–2720. PMLR, 2022.

Guangyu Chen, Yu Wu, Shujie Liu, Tao Liu, Xiaoyong Du, and Furu Wei. Wavmark: Watermarking for audio generation. *arXiv preprint arXiv:2308.12770*, 2023.

Yushen Chen, Zhikang Niu, Ziyang Ma, Keqi Deng, Chunhui Wang, Jian Zhao, Kai Yu, and Xie Chen. F5-tts: A fairytaler that fakes fluent and faithful speech with flow matching. *arXiv preprint arXiv:2410.06885*, 2024.

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.

Zhihao Du, Yuxuan Wang, Qian Chen, Xian Shi, Xiang Lv, Tianyu Zhao, Zhifu Gao, Yexin Yang, Changfeng Gao, Hui Wang, et al. Cosyvoice 2: Scalable streaming speech synthesis with large language models. *arXiv preprint arXiv:2412.10117*, 2024.

ElevenLabs. Speech to Speech. `https://elevenlabs.io/app/speech-synthesis/speech-to-speech`, 2024. Accessed: 2025-09-21.

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.

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

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

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*, pp. 13916–13932. PMLR, 2023.

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)*, pp. 6367–6371. IEEE, 2022.

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.

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.

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.

Chang Liu, Jie Zhang, Tianwei Zhang, Xi Yang, Weiming Zhang, and Nenghai Yu. Detecting voice cloning attacks via timbre watermarking. In *NDSS*, 2024.

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

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

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

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)*, pp. 1841–1844. IEEE, 2012.

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)*, pp. 5206–5210. IEEE, 2015.

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.

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: 2025-09-21.

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.

Reuters. Fact check: Video does not show joe biden making transphobic remarks, 2023. URL https://www.reuters.com/article/fact-check/idUSL1N34Q1IW. Accessed: 2025-09-21.

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, pp. 749–752. IEEE, 2001.

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.

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*, pp. 43180–43196, 2024.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

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

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*, pp. 4214–4217. IEEE, 2010.

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)*, pp. 6369–6373. IEEE, 2021.

Vocloner. Vocloner: AI Voice Cloning Tool. https://vocloner.com/, 2024. Accessed: 2025-09-21.

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.

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*, pp. 4006, 2017.

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.

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*, pp. 21–26, 2008.

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

LLM USAGE STATEMENT

Large Language Models (LLMs) were used exclusively as general-purpose writing assistants to improve readability and adjust formatting. They did not contribute to the research ideation, methodology, experimental design, analysis, or interpretation of results. All technical content and scientific contributions are solely the work of the authors.

A RELATED WORKS

A.1 PASSIVE DETECTION OF SPEECH TAMPERING

Audio editing process can generate artifacts or modify natural acoustic features within human speech. For example, frame offset (Yang et al., 2008), inconsistent noise (Pan et al., 2012), and even discontinued electric network frequency (Rodríguez et al., 2010; Esquef et al., 2014), 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 (Blue et al., 2022). Nevertheless, as the deepfake models evolve, relying solely on passive detection may not be sufficient against future attacks.

A.2 PROACTIVE PROTECTION OF SPEECH INTEGRITY

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 (Zakariah et al., 2018). However, hashing operations are too sensitive to tolerate common operations from regular users, such as audio compression and resampling, resulting in a high false alarm rate. Another method is fragile watermarking (Renza et al., 2018), 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. (2025) 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.

B SPEECHECK DESIGN AND OPERATION DEFINITIONS

B.1 OVERALL ALGORITHM

The training and verification procedures of SpeeCheck are summarized in Algorithm 1 and Algorithm 2, respectively.

B.2 DEFINITION OF BENIGN AND MALICIOUS OPERATIONS

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 its characteristics in Table 5.

Algorithm 1 SpeeCheck 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**
    $\mathcal{X}^{\text{benign}} \leftarrow \mathcal{T}_b(\mathcal{X}), \quad \mathcal{X}^{\text{malicious}} \leftarrow \mathcal{T}_m(\mathcal{X})$
    Step 1: Frame-level feature extraction
    $\mathcal{Z} \leftarrow \varepsilon(\mathcal{X})$
    Step 2: Multiscale feature summarization
    $\mathbf{h}_n \leftarrow \mathcal{F}(\mathcal{Z})$
    for $n = 1, \ldots, K$ **do**
      $w_n \leftarrow \frac{\exp(\phi(\mathbf{h}_n))}{\sum_{t=1}^{K} \exp(\phi(\mathbf{h}_t))}$
    end for
    $\mathbf{v}' \leftarrow \sum_{n=1}^{K} w_n \cdot \mathbf{h}_n$
    $\mathbf{v} \leftarrow \text{Proj}(\mathbf{v}')$
    Step 3: Contrastive fingerprint training
    Compute contrastive loss $\mathcal{L}_{\text{c}}$
    Update $\mathcal{F}$, ϕ, Proj via backpropagation
  end for
end for
Step 4: Binary fingerprint encoding
$\mathbf{b} \leftarrow \text{sign}(\tanh(\text{Proj}(\text{AttPool}(\mathcal{F}(\varepsilon(\tilde{X}))))))$
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**
  $\delta^{(n)} \leftarrow \text{WatermarkEmbedder}(\mathcal{X}^{(n)}, \mathbf{b}^{(n)})$
  $\tilde{\mathcal{X}}^{(n)} \leftarrow \mathcal{X}^{(n)} + \delta^{(n)}$
end for
$\tilde{\mathcal{X}} \leftarrow \text{Concat}(\tilde{\mathcal{X}}^{(1)}, \ldots, \tilde{\mathcal{X}}^{(N)})$

Algorithm 2 SpeeCheck 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

Table 5: Summary of audio operations.

Operation	Example	Implementation
Benign Operations		
Compression	Podcasts, news broadcasts, online meetings	`ffmpeg` (MP3 @ 128 kbps)
Reencoding	Saving or uploading audio files	`ffmpeg` (PCM 16-bit)
Resampling	Low-bandwidth communication	`Resample (torchaudio)`
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` (training), `Voice Changer` (testing)
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(128 kbps) or AAC (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. Note that integrating voice conversion models into the training pipelines is computationally expensive and time-consuming, making large-scale training impractical. To achieve a comparable effect with lower overhead, during the training phase, we apply pitch shifting for speaker identity modification (e.g., +4 semitones) using SoX effects, implemented via `torchaudio.sox_effects.apply_effects_tensor`. This modification introduces perceptual changes to voice characteristics, effectively creating negative samples for learning to distinguish speaker identity. In the testing phase, we validate SpeeCheck's performance on a separate set of audio manipulated by a state-of-the-art commercial voice changer tool from ElevenLabs (ElevenLabs, 2024).

Text-to-Speech. We synthesize speech from text using a pre-trained text-to-speech (TTS) model, YourTTS (Casanova et al., 2022), 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.

B.3 Explanation of Malicious Tampering over Different Granularities

Table 6 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 6: 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 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.

C Experimental Setup and Extended Results

C.1 Implementation Details

To supplement Section 4.1, we provide a detailed description of the model architecture and training configuration.

Model. We use the pretrained Wav2Vec2.0 Base model[4] 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[5] 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. SpeeCheck 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 2 NVIDIA A100 GPUs using distributed data parallelism.

Table 7: Examples from RWSID with corresponding editing operations

Sentence	**Editing Operation**
The board has decided they can not approve the new budget.	Deletion / Silencing ("not")
Our analysis shows this investment is not a secure option.	Deletion / Silencing ("not")
Based on the evidence, the suspect is innocent.	Substitution → "guilty"
Based on the evidence, the suspect is guilty.	Substitution → "innocent"
I never said she stole the company's data.	Reordering
I never said she stole the company's data.	Voice Conversion (AI)
We will begin the product launch immediately.	Replacement → "delay"
We will delay the product launch immediately.	Replacement → "begin"
I believe it is a good idea, but we need more time.	Splicing
This is authentic audio, not deepfake.	Text-to-Speech (AI)

C.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. To comprehensively evaluate the effectiveness of SpeeCheck in real-world scenarios, we construct a **Real-World Speech Integrity Dataset** (RWSID). This dataset comprises recordings from 10 volunteers with diverse demographic backgrounds (including multiple races and sexes). Each participant read 8 prepared speeches (see Table 7). All audio files are converted to WAV format and resampled to 16 kHz.

Preprocessing. We randomly sample 10,000 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.

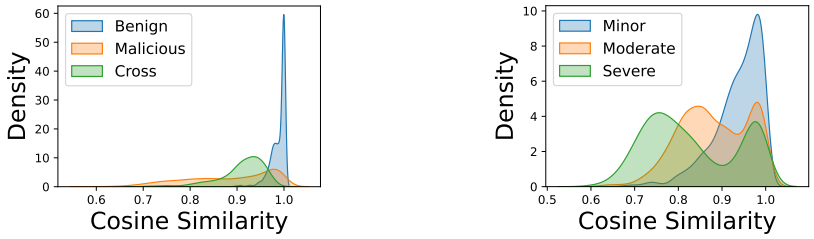
For each valid utterance, we generate two sets of augmented variants for contrastive learning: (i) Benign Augmentations: These are modifications that preserve both speaker identity and semantic content. (ii) Malicious Augmentations: These include tampering operations intended to alter speaker identity and semantic content. The details can be found in Appendix B.2.

Evaluation Metrics. 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:

[4] `https://github.com/facebookresearch/fairseq/blob/main/examples/wav2vec`
[5] `https://github.com/facebookresearch/audioseal`

- True Positive Rate (TPR): $\text{TPR} = \text{TP}/(\text{TP} + \text{FN})$
- False Positive Rate (FPR): $\text{FPR} = \text{FP}/(\text{FP} + \text{TN})$
- True Negative Rate (TNR): $\text{TNR} = \text{TN}/(\text{TN} + \text{FP})$
- False Negative Rate (FNR): $\text{FNR} = \text{FN}/(\text{FN} + \text{TP})$
- Equal Error Rate (EER): The error rate at the decision threshold where FPR = FNR.
- Area Under the Curve (AUC): The area under the receiver operating characteristic curve.

C.3 Similarity Distribution using MFCC Feature



(a) (b)

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

To complement the observations in Section 2.3, 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.

C.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[6], `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[7], `speechbrain/spkrec-ecapa-voxceleb`.

As shown in Table 8, benign operations (e.g., compression, recording, resampling, noise suppression) result in low WER ($\leq$8.24%) and high identity similarity ($\geq$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.

[6] `https://github.com/facebookresearch/fairseq/blob/main/examples/wav2vec`
[7] `https://huggingface.co/speechbrain/spkrec-ecapa-voxceleb`

Table 8: 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**

C.5 Results of Fine-Grained Malicious Operations Rejection

We report the detection performance of SpeeCheck on fine-grained malicious operations across varying degrees of tampering severity, as shown in Table 9.

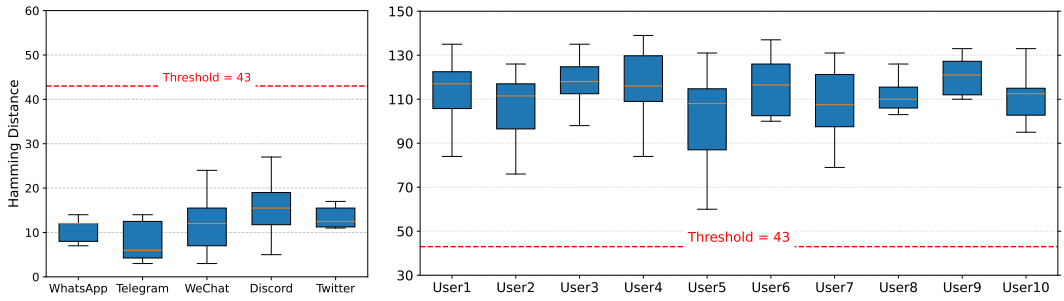
Table 9: Results of fine-grained malicious operation rejection on VoxCeleb and LibriSpeech.

Operation	VoxCeleb				LibriSpeech				Semantic	Identity
	TNR	FNR	AUC	EER	TNR	FNR	AUC	EER		
Deletion (minor)	100.00	2.20	99.92	0.60	100.00	5.20	99.77	1.20	✗	✓
Deletion (moderate)	100.00	2.80	99.96	0.40	100.00	5.20	99.84	0.80	✗	✓
Deletion (severe)	100.00	3.40	99.99	0.20	100.00	4.40	99.76	0.80	✗	✓
Splicing (minor)	100.00	2.40	99.99	0.40	100.00	4.20	99.57	1.30	✗	✓
Splicing (moderate)	100.00	2.40	99.99	0.40	100.00	4.80	99.79	0.40	✗	✓
Splicing (severe)	100.00	2.60	99.98	0.20	100.00	5.80	99.72	0.50	✗	✓
Silencing (minor)	98.00	3.60	99.73	3.20	86.60	6.00	97.34	8.70	✗	✓
Silencing (moderate)	99.60	2.40	99.90	1.80	97.20	4.20	98.82	4.10	✗	✓
Silencing (severe)	99.60	2.80	99.86	1.80	98.80	6.40	99.12	4.00	✗	✓
Substitution (minor)	88.40	3.00	98.75	6.10	72.20	4.40	95.50	11.80	✗	✓
Substitution (moder.)	99.20	3.20	99.62	2.60	92.80	5.20	98.24	5.50	✗	✓
Substitution (severe)	100.00	3.00	99.80	1.20	100.00	4.80	99.27	2.70	✗	✓
Reordering	97.60	2.60	98.44	2.60	99.20	5.80	99.08	1.70	✗	✓
Text-to-speech	100.00	0.00	100.00	0.00	100.00	0.00	100.00	0.00	✗	✓
Voice conversion	100.00	3.00	99.93	0.50	100.00	5.40	99.64	0.50	✓	✗
Overall	98.83	2.63	99.72	1.47	96.45	4.79	99.05	2.93	-	-

C.6 Evaluation in Real-World Scenario

To validate SpeeCheck's performance in practical settings, we conducted evaluations on the RWSID dataset (described in Appendix C.2). Example recordings and verification results are available on our demo page.[8] We then designed two evaluation scenarios to simulate real-world challenges:

[8] https://speecheck.github.io/SpeeCheck/



(a) (b)

Figure 7: Hamming distance distributions for real-world scenarios: (a) benign social media distribution and (b) malicious tampering.

Benign Distribution. To assess SpeeCheck's robustness under practical distribution scenarios, the protected audios were uploaded to widely used social media platforms (including WhatsApp, Telegram, WeChat, Discord, and Twitter) and subsequently downloaded after platform-side preprocessing. These steps reflect realistic distribution pipelines where compression, reencoding, and noise suppression may be applied.

Malicious Tampering.: To evaluate its sensitivity to sophisticated attacks, we performed fine-grained edits manually, including deletion, splicing, silencing, substitution, and reordering. Moreover, we tested commercial platforms for voice conversion (ElevenLabs, 2024) and text-to-speech synthesis (Vocloner, 2024).

Figure 7a shows that the Hamming distances of audios redistributed via social media consistently remain below the detection threshold, confirming SpeeCheck's robustness against real-world distribution. In contrast, Figure 7b demonstrates that all malicious edits yield Hamming distances above the threshold across all 10 users, indicating its reliable sensitivity to real-world tampering.

C.7 Evaluation on Different Lengths of Speech

To evaluate SpeeCheck on longer audio recordings, we tested speech samples ranging from 20 seconds to 10 minutes. As shown in Table 10, the system performs well across all lengths. While performance gradually degrades with increasing duration, the EER rises from 1.57% at 20 seconds to 8.41% at 10 minutes, the overall detection remains robust, with the AUC consistently above 96%.

Table 10: Performance of SpeeCheck under different speech durations.

Speech Duration	TPR	FPR	AUC	EER
20s	98.75	1.65	99.69	1.57
60s	92.50	4.88	98.39	5.87
5m	94.87	7.05	98.02	6.09
10m	90.95	7.76	96.04	8.41

C.8 Real-Time Evaluation

Table 11: Real-time performance of SpeeCheck.

Process	Real-Time Coefficient (RTC)
Protection	0.02×
Verification	0.03×

We evaluate computational efficiency using the Real-Time Coefficient (RTC), defined as the ratio of processing time to audio duration. As shown in Table 11, both protection and verification achieve RTC values well below 1×, confirming the practicality of SpeeCheck for real-time use.

C.9 WATERMARKED SPEECH QUALITY

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) (Rix et al., 2001) ranges from 1.0 (poor) to 4.5 (excellent), and reflects perceived speech quality. (3) Short-Time Objective Intelligibility (STOI) (Taal et al., 2010) 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 12, our proposed SpeeCheck 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.

Table 12: Audio quality metrics.

Methods	**SI-SNR**	**PESQ**	**STOI**	**LSD**
SpeeCheck	25.14	4.28	0.998	0.111

C.10 ABLATION STUDIES

To access the contribution of SpeeCheck's core modules, we conduct ablation studies on three key components: the multiscale feature extractor, the attentive pooling module, and the contrastive learning objective. We evaluate the multiscale feature extractor by removing it and using the direct output of BiLSTM. For temporal pooling, we substitute attentive pooling with average pooling. Finally, we compare the InfoNCE loss (Oord et al., 2018) with widely-used Triplet Loss (Schroff et al., 2015).

As shown in Table 13, each module contributes to the overall performance. 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 13: Ablation study on feature extractor, temporal pooling scheme, and loss function.

Method Variant	**TPR**	**FPR**	**AUC**	**EER**
SpeeCheck (Multiscale $\rightarrow$ w/o Multiscale)	94.48	5.26	98.50	5.29
SpeeCheck (AttentivePooling $\rightarrow$ AvgPooling)	93.80	6.25	98.42	6.22
SpeeCheck (InfoNCEloss $\rightarrow$ TripletLoss)	85.94	1.46	95.77	10.73
SpeeCheck	**98.70**	**0.85**	**98.89**	**0.92**

C.11 Visualization of Multiscale Feature Extraction

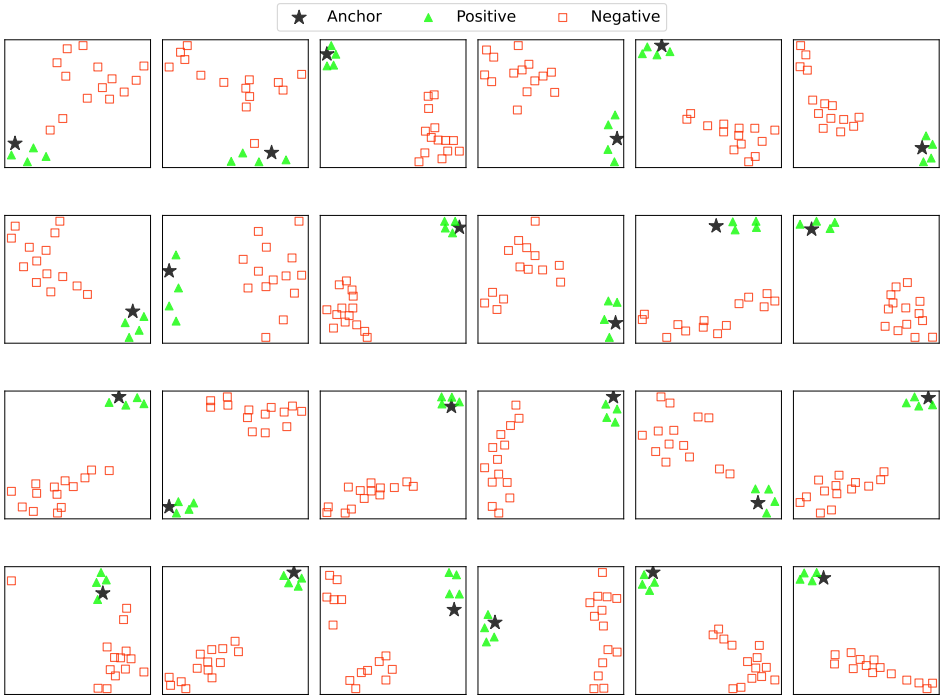


Figure 8: t-SNE visualizations of speech samples (after training).

D Discussion

While SpeeCheck provides a new paradigm for self-contained speech integrity verification, we acknowledge several limitations that can be improved in future research: 1) SpeeCheck's robustness is limited concerning certain operations like time-stretching (speeding up/slowing down) and Voice Activity Detection (VAD). These operations are sometimes not malicious, but they can inherently alter pitch, tempo, or meaningful phonemes in the speech. Our framework prioritizes a security-first design, we conservatively treat modified speech as unreliable. But extending training with such operations, or integrating more robust watermarking schemes, could improve the applicability in the future. 2) SpeeCheck is optimized for speech durations between 2 and 20 seconds. For very short clips, the embedding and watermark extraction may become unstable. However, such clips often lack meaningful semantic content, making them less critical targets for tampering. For very long audio, although we did not explicitly train on durations beyond 20 seconds, SpeeCheck still exhibited reasonable generalization. A practical solution is to segment longer recordings into multiple 20-second chunks, protect and verify the content in controllable chunks. 3) Current SpeeCheck is tailored for speech integrity verification rather than general audio (e.g., music, environmental sounds). This choice is motivated by the fact that Speech is particularly vulnerable to tampering and can significantly impact social trust and social stability. Generalizing the system to broader audio domains would be a promising direction for future work, for instance, emphasizing perceptual fidelity, spectral consistency, and artistic style preservation. 4) SpeeCheck provides an utterance-level tampering detection, but does not localize the tampered region. This is because SpeeCheck summarized each speech into a concrete digest. Extending SpeeCheck with temporal localization could be a promising future direction.