Security Analysis of the audiowmark Scheme

Karen Guo Selinna Lin Jolin Yang Angela Zhang
{karguo, selinna, yangj216, ayz26} @ mit.edu

Abstract

Audio watermarking embeds inaudible, machine readable data into audio signals. The watermarking scheme, audiowmark, is an open source project embedding a 128-bit payload protected by a private AES-based key into an audio file, claiming inaudibility, robustness to lossy compression, and security without the private key.We identify three weaknesses: the lack of internal authentication for an existing watermark, additive watermark structure in log-power space, and synchronization frames that depend only on the private key. We evaluate four attacks: (1) layered overlay producing conflicting ownership claims, (2) repeated overlay suppressing the original watermark, (3) watermark reconstruction from clean and watermarked audio pairs, (4) watermark removal from averaging same-key watermarked files. Finally, we propose three defenses: (1) a per file random nonce, (2) a perceptual hash binding, and (3) a commitment with MAC ownership protocol.

1. Introduction

The increase in digital audio content has fundamentally changed how music, podcasts, and other media are created and distributed. While broadening access, it also made copyright infringement and unauthorized distribution much easier. Audio watermarking has emerged as a key solution to address these concerns, enabling copyright protection, content authentication, and broadcast monitoring by embedding an imperceptible identifier into an audio signal. This identifier can be later extracted to verify authenticity, prove ownership, and track distribution [4]. However, for these applications to hold up in practice, the underlying watermarking scheme must be secure. If an attacker can effectively remove or forge a watermark, the entire chain of trust these systems depend on collapses.

The open source (GPL)[1] audio watermarking scheme, audiowmark, embeds an inaudible, 128-bit digital watermark into audio files, designed to be robust against lossy compression [5]. Its primary use cases are copyright protection, proof of ownership, and tracking leaks of audio content, such as music or voice recordings. It has seen adoption beyond individual use, serving as a reference point in academic benchmarking and as a source of inspiration for subsequent watermarking schemes such as WavMark, an AI-based audio watermarking tool [1]. The watermarking scheme claims security through a private key mechanism, through which knowledge of the algorithm alone is insufficient to read or modify a watermark without the corresponding key [6].

In this paper, we analyze the scheme's security, identify a set of structural vulnerabilities, and demonstrate practical attacks exploiting them. Finally, we propose countermeasures to address the identified weaknesses.

2. audiowmark

The open-source audio watermarking scheme, audiowmark, embeds a 128-bit message into audio that is inaudible to humans and allows messages to be extracted later without access to the original audio. The scheme is built around a private 128-bit AES key K, which determines watermark placement and encoding throughout the pipeline.

The scheme contains four relevant properties [6]:

1. **Inaudibility**: The added watermark is imperceptible to listeners.
2. **Robustness**: The watermark survives lossy compression such as MP3.
3. **Blind Decoding**: Detection does not require the original audio.
4. **Security**: Only users with the correct key can read or modify the watermark.

[1] The GNU General Public License is a widely used free, open-source software license.

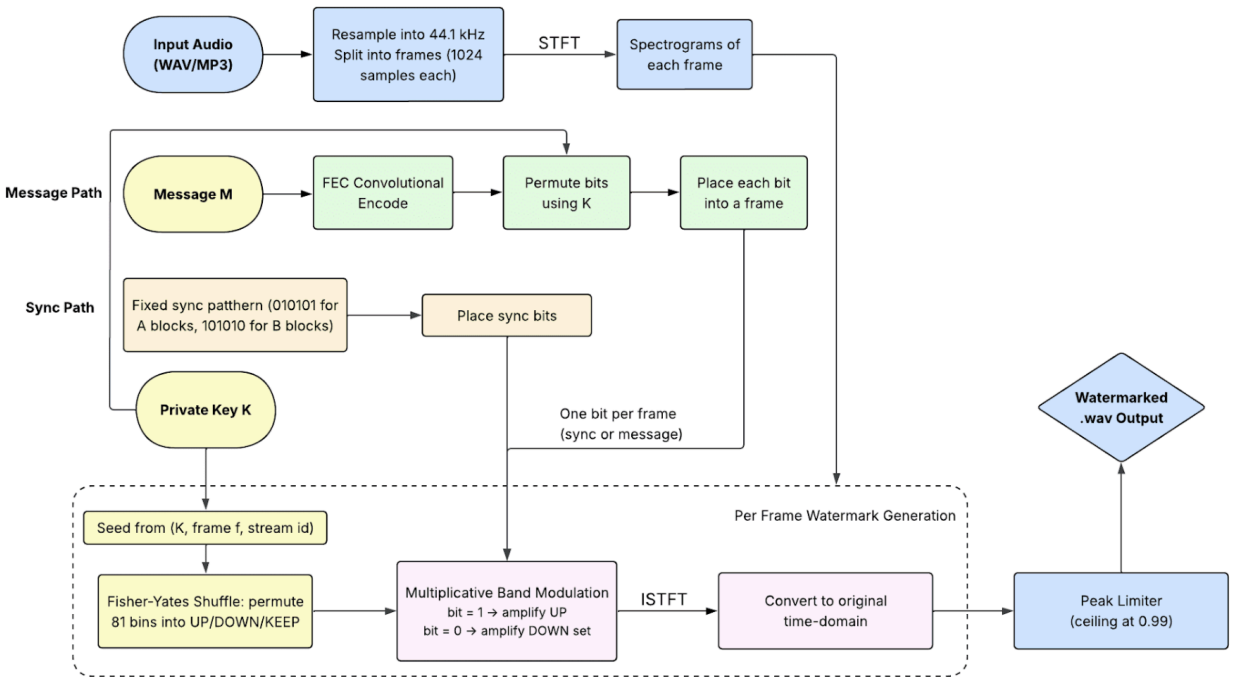

Figure 1. Block diagram of the audiowmark embedding pipeline.

2.1. Embedding Algorithm

The embedding process transforms a 128-bit payload into inaudible frequency-domain perturbations spread across the audio. A private key K controls the pseudorandom placement and encoding of the watermark.

1. The input audio is resampled to 44.1 kHz stereo, divided into 1024-sample frames, and transformed into the frequency domain using the FFT[2].

2. The payload is then convolutionally encoded to add redundancy for robust recovery after compression or degradation.

3. The private key K seeds several AES-based pseudorandom streams that determine the permutation and interleaving of encoded bits, the placement of synchronization and payload data across frames, and the assignment of spectral modification within each frame.

4. The encoded payload bits are combined with highly redundant synchronization patterns and organized into 2 block types: A-blocks and B-blocks

5. For each frame, selected FFT bins are partitioned into UP, DOWN, and KEEP groups using a Fisher–Yates shuffle[3] derived from K, the frame index, and stream-specific randomness. Depending on the assigned bit, either the UP or DOWN bins are amplified while the other is attenuated.

6. The perturbation is applied multiplicatively to the magnitude of the selected FFT coefficient. The frames are then transformed back into the time domain, passed through a peak limiter to keep samples within the valid range, and written to an output audio file.

2.2. Extracting Algorithm

The extraction process detects and, if present, extracts the 128-bit payload embedded in an audio file using the same key used during embedding.

1. The suspect audio file is resampled to 44.1 kHz and divided into 1024-sample frames. Each frame is transformed into the frequency domain using the FFT, reproducing the same spectral representation used during embedding.

[2]Fast Fourier Transform, an efficient algorithm for computing the Discrete Fourier Transform, converting signals between time, space and frequency domains

[3]Algorithm for generating a uniformly random permutation of a finite sequence

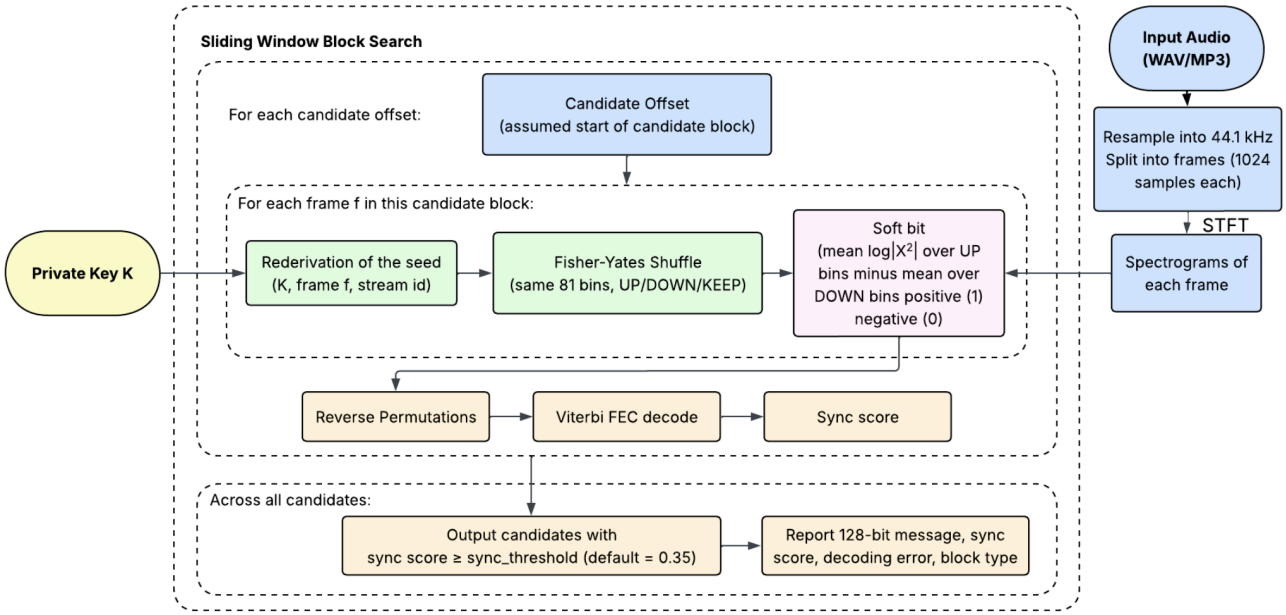

Figure 2. Block diagram of the audiowmark detection pipeline.

2. The decoder searches over all A-block and B-block start positions, reconstructing the sync-bit locations and UP/DOWN band assignments using K. For each frame, the difference between mean log-power over the UP and DOWN bins gives a soft bit: its sign indicates the recovered bit, its magnitude the decoder's confidence.

3. The extracted bit stream is deinterleaved and reordered using the same key-dependent permutations applied during embedding, reversing the randomized placement and interleaving process performed by the encoder.

4. The reordered bits are passed through a Viterbi decoder (the standard algorithm for decoding convolutional codes). This outputs the original 128-bit payload plus an error metric indicating how confident the decoding is.

5. The decoder then computes the sync score by checking if the candidate's sync frames match the known fixed pattern. A higher sync score means that the candidate is more likely to be a real watermark block.

3. Vulnerabilities

3.1. Absence of Internal Ownership Authentication

The audiowmark scheme does not perform any check for the presence of an existing watermark before embedding. The scheme blindly applies its embedding process to all incoming audio regardless of whether a watermark is already present. Since detecting an existing watermark only relies on the presence of the corresponding private key K_V, an attacker with access to any watermarked file can create a new key K_A and embed a second watermark on top of the original one.

This opens two forms of attack. An attacker can embed their own watermark to claim false ownership of content, or progressively degrade the Signal-to-Noise Ratio (SNR) of the original watermark such that it is no longer detectable, all without knowledge of K_V.

3.2. Additive in Log-Power Space

Although audiowmark embeds multiplicative perturbations in FFT magnitudes, they become additive in the log-power space:

$$|X| = |C| \cdot g \implies \log|X|^2 = \log|C|^2 + W$$

where C is the clean audio STFT, X is the watermarked audio STFT, and W is the watermark pattern.

As described in Section 2.1, embedding scales each affected FFT bin's magnitude by a factor g derived from the payload and key K. While g depends on C, this dependence primarily controls watermark strength so W's locations remains consistent across every file watermarked with the same key.

This creates a structural leakage. Since the watermark is shared across files while the audio content

varies independently, an attacker with access to multiple files watermarked under the same key K could average their log-power spectrograms. We predicted that as the number of files N increases, the audio distributions average out with their variance shrinking at a rate of $\frac{1}{\sqrt{N}}$, while preserving the shared watermark.

3.3. Sync Frames and Data Frames

Each watermark block contains sync frames and data frames. A sync frame carries a fixed alternating bit pattern for "A" blocks (010101) and "B" blocks (101010). This pattern is identical in every audiowmark output and is used by the detector to find block boundaries. Placement of these patterns is independent of the user's message, but is deterministic in K. A data frame carries the payload, or FEC[4](message), and placement pattern depends on both K and the user's message M.

For the same key K, the data-frame sign patterns can vary as the data pattern depends on M while the sync-frame sign patterns are identical across audio files. This means that a "per-file unique message" defense partially defeats the data frame averaging attack suggested in Section 3.2, but fails to protect against sync frame averaging. The sync portion is a separate keystream-reuse vulnerability that survives the message-diversity defense, as the adversary recovers the per-frame sign assignments at sync-frame positions without ever recovering K itself.

4. Attacks and Results

4.1. Metrics

- **Sync Score**: A quantity in [0,1] measuring how well the recovered sync bits match the expected pattern with higher values indicating a better match. We use the default detection threshold of 0.35.
- **Decoder Bit Error Rate**: The fraction of uncertain or incorrectly decoded payload bits reported by the decoder. With 0.5 as a threshold, 0.0 means every bit is decoded confidently, while 0.5 is no better than random guessing.
- **Signal-to-Noise Ratio (SNR)**: A logarithmic measure of audio fidelity.

$$\mathrm{SNR}_{\mathrm{dB}} = 10 \log_{10} \left(\frac{\sum x_{wm}[n]^2}{\sum \left(x_{wm}[n] - x_{attack}[n] \right)^2} \right)$$

where x_{wm} is the original watermarked audio and x_{attack} is the audio produced by the attack. Higher SNR values mean that the audio is more recognizable. We use 10 dB as the threshold for preserved audio, as attacks that degrade the audio are trivial to construct.

4.2. Watermark Overlay Attack

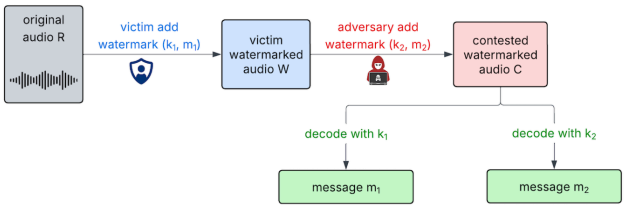

Figure 3. Diagram of the watermark overlay attack.

4.2.1 Threat Model

The attacker has access to one or more watermarked audio files and can run the embedding algorithm with their own keys and messages, but does not know the victim's secret key.

4.2.2 Goal

The attacker's goal is to overlay their own message into the file, decodable with their key, while keeping the original victim's watermark and audio file structure intact. We target the vulnerability in Section 3.1.

4.2.3 Attack

To evaluate the impact of the absence of internal ownership authentication, we implemented a simple multi-watermark attack.

The victim first embeds a watermark with their own hashed K_1 and payload M_1. Then, the attacker then embeds a second watermark with key $K_2 \neq K_1$ and payload $M_2 \neq M_1$. Then, we try to decode with either key. We call the attack a success if both messages can be recovered with their respective keys above a sync score of 0.35 and the original audio is preserved via manual listening and checking SNR $\geq$ 10db.

We verified across 10 audio samples that the resulting file could be successfully decoded using either key. We also observed that layering three or more watermarks begin to corrupt earlier messages, whereas two overlaid watermarks almost always remained decodable.

[4]Forward Error Correction, a digital signal processing technique that enhances data reliability by adding redundant bits to data

This shows that multiple independent parties can produce valid ownership claims over the same audio file without ever accessing the victim's key.

4.3. Watermark Overriding Attack

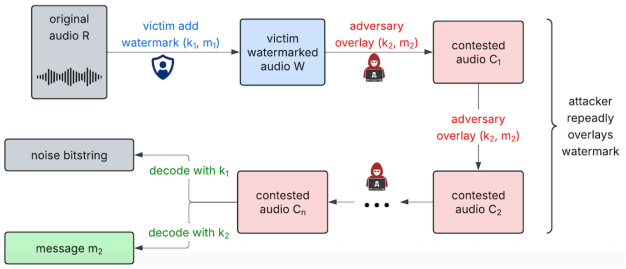

Figure 4. Diagram of the watermark override attack.

4.3.1 Threat Model

The attacker has access to one or more watermarked audio files and can run the embedding algorithm with their own keys and messages, but does not know the victim's secret key.

4.3.2 Goal

The attacker's goal is to overlay their own message into the file, decodable with their key, while keeping the original victim's watermark and audio file structure intact. We target the vulnerability in Section 3.1.

4.3.3 Attack

This attack takes the previous attack a step further and attempts to destroy the original watermark while keeping the attacker's watermark and the audio file intact. The attacker repeatedly encodes their own key and message on top of a victim's file. After each layer, we check if the original message is still decodable. The attack is defined as successful if the original message is no longer recoverable (i.e. not the top message recovered or a very low sync score) while the attacker's watermark message is decodable and the audio file is still valid via manual listening and ensuring SNR $\geq$ 10db.

We observed that after roughly three overlays, the victim's original watermark becomes extremely corrupted, preventing the detector from reliably recovering it. However, the attacker's watermark still remains detectable. This demonstrates how a lack of authenticated or protected ownership allows an attacker to repeatedly re-watermark a file to erase evidence of an original owner and claim false ownership.

4.4. Watermark Reconstruction Attack

4.4.1 Threat Model

The adversary has access to one or more clean and watermarked audio pairs generated using the same embedding key K and payload M. The adversary does not know the secret key K, but can observe both the original and corresponding watermarked audio.

4.4.2 Goal

The adversary's goal is to estimate the reusable watermark structure from these pairs and transfer it onto previously unseen clean audio files such that the forged audio successfully decodes under the victim's key K. We target the vulnerability in Section 3.2.

4.4.3 Attack

The attack first takes pairs of clean and watermarked audio (R_i, X_i), generated with the victim's secret key K and message M. The attack can be performed with N pairs, where $N \geq 1$. The audiowmark scheme applies a multiplicative perturbation in the STFT magnitude domain, so in log-power space the watermark becomes approximately additive as described in Section 3.2.

$$\log |\mathrm{STFT}(X_i)|^2 \approx \log |\mathrm{STFT}(R_i)|^2 + W_{K,M}$$

where $W_{K,M}$ is the watermark pattern determined by the key K and message M. The attacker estimates the watermark residual from each known pair:

$$W_{est,i} = \log |\mathrm{STFT}(X_i)|^2 - \log |\mathrm{STFT}(R_i)|^2$$

Finally, they can average the residuals for the final estimate:

$$W_{est} = \frac{1}{N} \sum_{i=1}^{N} W_{est,i}$$

A larger N reduces noise from the underlying audio content and improves the estimate of the reusable watermark structure. After obtaining the estimated watermark, we can forge the watermark onto an unseen clean target audio T. The attacker then computes:

$$\log |\mathrm{STFT}(T')|^2 = \log |\mathrm{STFT}(T)|^2 + W_{est}$$

The attacker converts the modified spectrogram back to the time domain using inverse STFT, producing a candidate forged audio file T'. The attack succeeds if audiowmark decoding on T' with the victim's key K outputs the victim's message M with sync score above 0.35 and decoder bit error less than 0.5.

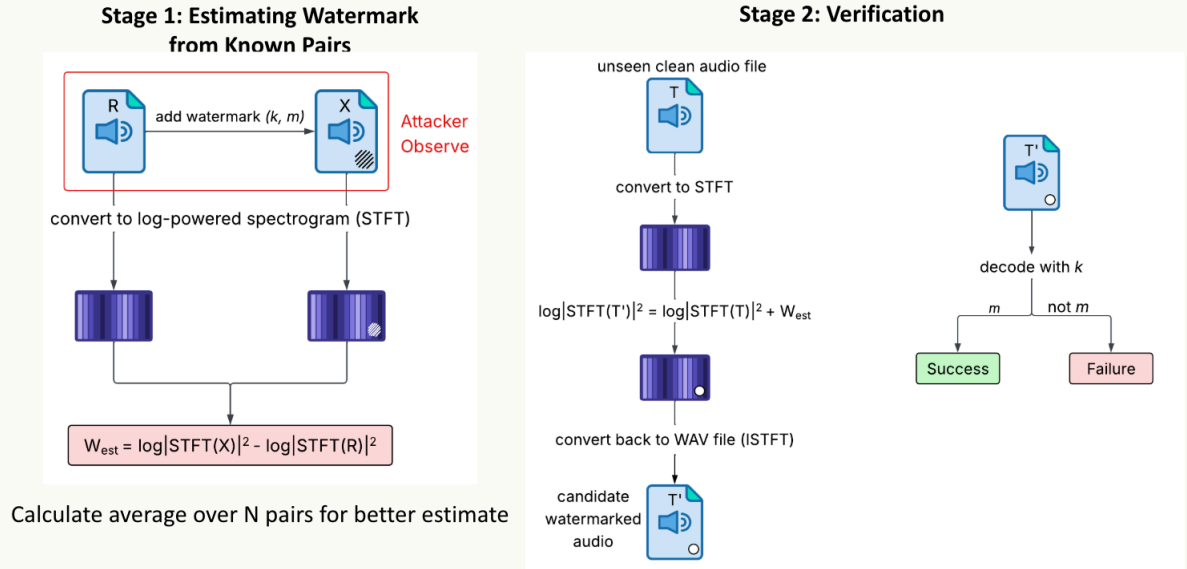

Figure 5. Diagram of watermark reconstruction attack.

4.4.4 Evaluation

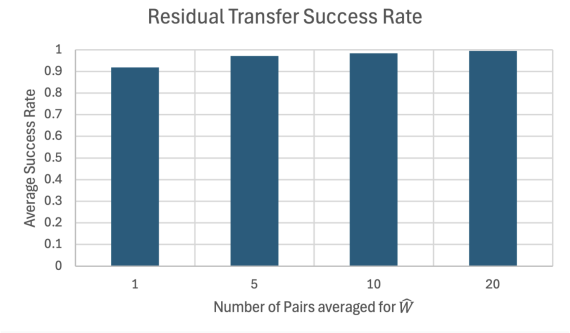

Figure 6. Watermark Reconstruction/Transfer Attack Success Rate

Figure 6 shows the results for the attack using different numbers of audio pairs. We averaged the estimated watermark over each pair, then applied the resulting W_{est} to 60 previously unseen clean audio files. We repeated the experiment six times for each number of pairs, using a random permutation to determine grouping, and reported the average success rate across runs.

We observe that even with only a single pair, the attack already achieves a very high success rate of approximately 90%. The estimate becomes more accurate as the number of pairs increases and approaches nearly 100% by around 20 pairs. This shows that audiowmark contains a highly reusable and transferable structure that can be reliably reconstructed if attackers observe clean and watermarked audio pairs together. This allows the attacker to forge the signature of the victim and apply it to any other clean audio files for potential framing purposes.

4.5. Watermark Removal Attack

4.5.1 Threat Model

We relax the threat model from Section 4.4 by assuming that the attacker only has access to publicly released watermarked audio files generated using the same embedding key K but not to clean source files. The attacker does not know K.

4.5.2 Goal

The adversary's goal is to remove the watermark from a target file given only a corpus of N audio files watermarked under the same key K. This attack targets the vulnerabilities in Sections 3.2 and 3.3.

4.5.3 Attack

First, the client watermarked N audio files with key K and message M, modeling publicly released watermarked content available to an adversary. Then the adversary converts each watermarked file to a log-power spectrogram, averages across all N files, and subtracts the frequency-axis median to estimate the watermark W_{est} without recovering K. Since the watermark is deterministic in K and frame index, averaging isolates the K-dependent pattern while content cancels. Next, the adversary subtracts a scaled W_{est} from the target

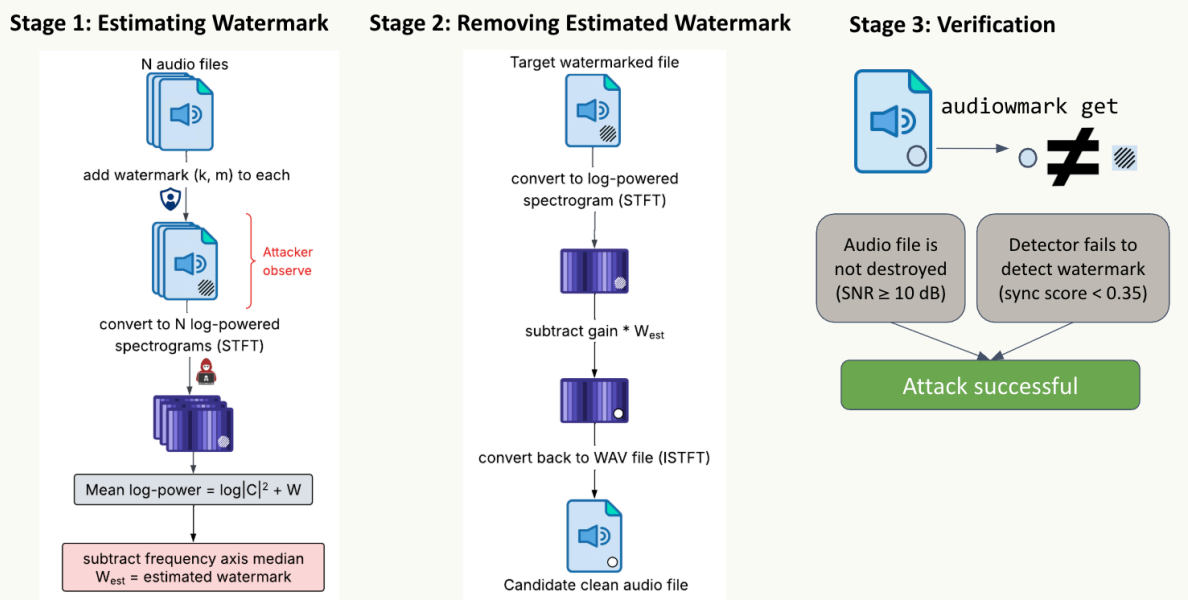

Figure 7. Diagram of the watermark removal attack.

file's log-power spectrogram, and then invert it back to audio. This should now give us a theoretically clean file if the watermark was successfully removed. To verify this, we measure SNR between the cleaned and original watermarked audio. A ratio $\geq$ 10 dB means that the audio is still recognizable. We then run audiowmark get on the cleaned file: if the sync score falls below 0.35, the detector has failed. The attack succeeds only if both conditions hold.

4.5.4 Evaluation

We swept across values of N (the number of watermarked files the adversary has) and gain (the amplitude scaling applied to W_{est} before subtraction), running the attack against a pool of 50 target files per condition. We ran the sweep under two scenarios: the same-message case, where every watermarked file carries the same M, and the per-file-message case, where each file carries a distinct m_i.

As predicted, increasing N improves the adversary's estimate of W and raises the attack success rate (Figure 8). In the same-message scenario we observed a peak success rate of 12% at $N = 80$, gain $= 2$. In the per-file-message scenario the peak dropped to 6% at $N = 160$, gain $= 2.5$. At higher gains, the attack is less effective since subtracting a larger W_{est} damages the audio enough to push SNR below the 10 dB threshold even when a watermark detection fails.

The reason for this gap is the data vs. sync asymmetry we identified in Section 3.3. The sync sign pattern

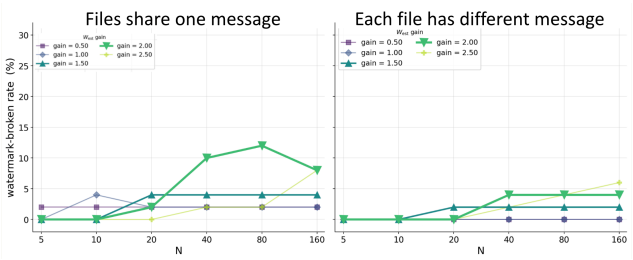

Figure 8. Watermark removal attack success rate as a function of corpus size N and subtraction gain, across same-message (left) and per-file-message (right) scenarios.

depends only on K, so it survives averaging in both scenarios and both conditions therefore show similar sync-score drops. But the data sign pattern depends on both K and M. In the same message case, the data pattern is shared across files and is recovered by averaging just like the sync pattern, allowing the adversary to scramble the message bits. In the per-file-message case, the data sign patterns differ across files and cancel during averaging, so the data bits remain decodable even when the sync detector fails. This shows up clearly in Figure 9, where the per-file-message points cluster below the 0.5 error threshold (bits still recoverable) while the same-message points cluster above it (bits scrambled). Both sets span similar sync-quality ranges, but the visible separation along the y-axis shows the asymmetry lies in the decoder bit-error rate.

This asymmetry has an important defense implication. A naïve fix is to use a fresh message per file, which

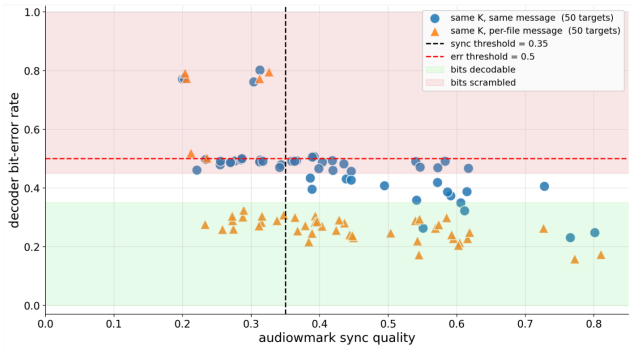

Figure 9. Per-target sync score versus decoder bit-error rate for 50 attacked files in each scenario. The dashed lines mark the sync threshold (0.35) and bit-error threshold (0.5); the shaded bands indicate where bits remain decodable (green) versus scrambled (red).

defeats data-frame averaging but does not address the sync-frame keystream reuse. The watermark detector can still fail since the sync portion is averaged away, even though the message bits are protected. A complete fix requires per-file freshness in K's derivation, and not just message diversity in the files.

5. Proposed Defenses

We propose three defenses that address the vulnerabilities mentioned in Section 3. Each defense modifies the encoding and decoding interfaces while leaving the underlying spectral embedding largely intact and preserving audio quality.

5.1. Per-File Random Nonce

Our first defense targets the keystream reuse vulnerability described in Section 3.2. During encoding, the embedder sample a fresh random nonce $R \in \{0,1\}^{128}$ and uses both K and R to derived all pseudorandom streams that audiowmark currently derives from key K alone.

The embedding process becomes:

$$(C, K, M) \rightarrow X = f(C, K, M, R) \rightarrow \text{output}(X, R)$$

where R is stored alongside the watermarked file for decoding. Decoding flow then becomes:

$$\begin{aligned}(X, K, R) &\rightarrow M = f^{-1}(X, K, R) \\ &\rightarrow \text{output}(M,\ \text{sync},\ \text{error},\ \text{type})\end{aligned}$$

With a fresh nonce R per file, the placement of UP, DOWN, and KEEP bins chances for every watermarked file. As a result, averaging log power spectrograms across files no longer reveals a reusable watermark estimate as both the synchronization and payload sign patterns become uncorrelated across files. This prevents the forgery attack in Section 4.4 and the removal attack in Section 4.5.

The primary tradeoff is that R must be returned along with the audio introducing an additional storage and communication requirement.

5.2. Perceptual Hash

Our second defense incorporates a perceptual hash. Unlike a standard cryptographic hash, a perceptual hash produces similar hashes for perceptually similar audio signal[5] [2]. By deriving watermark placements from audio content itself, the watermark pattern becomes file-dependent and cannot be reused across uncorrelated files.

Let H denote a perceptual hash. During encoding, the embedder computes $\varphi_C \leftarrow H(C)$ and uses φ_C with K and M to derive the pseudorandom streams, $X = g(C, K, M, \varphi_C)$. During decoding, the hash is recomputed from the given audio, $\varphi_X \leftarrow H(X)$ and decoded via $m = g^{-1}(X, K, \varphi_X)$. Successful decoding requires the perceptual hash to remain stable under watermark embedding and common audio transformations so φ_X yields the same pseudorandom streams as φ_C.

Because φ is a function of the cover, both sign pattern and frequency-bin placement vary across files even when K and M are shared. This breaks the forgery and averaging attacks in Section 4 because the attacker cannot transfer watermark estimates learned from training files onto unseen files with a different perceptual hash.

Unlike the nonce defense, no external side channel is required because φ can be recomputed directly from the watermarked audio. However, the defense depends on the stability of the perceptual hash. If the hash is changed under watermark embedding or normal audio transformations, decoding reliability may fail.

5.3. Commitment Scheme

Our third defense addresses the lack of ownership authentication. We propose augmenting audiowmark with a cryptographic commitment scheme and message authentication code (MAC).

During embedding, the owner computes a commitment over their identity, the payload message M, a hash of the audio, a timestamp t, and a random value r used to ensure binding and hiding [3].

$$C = \text{Commit}(\text{owner}, M, H(\text{audio}), t; r)$$

A MAC T can then be computed over C using the owner's private key K. The distributed output includes

[5] audio that would sound nearly identical to a human observer

the watermarked file together with (C, T, r). To verify ownership, a verifier recomputes the MAC using K and validates the commitment to recover the owner identity and timestamp.

This defense does not prevent an attacker from embedding a second watermark at the signal level, but provides a cryptographic proof of prior ownership that can be used to dispute fraudulent claims after the fact.

6. Conclusion

Our analysis shows that audiowmark's robustness against compression and signal degradation does not necessarily imply security against adversarial manipulation. In audiowmark, a reusable key-dependent structure creates unintended leakage across files, enabling attackers to forge ownership claims, override existing watermarks, and estimate watermark structure without the key itself.

The proposed defenses illustrate that preventing these attacks requires introducing randomness or external cryptographic authentication. A secure watermarking scheme should be designed with both signal-processing robustness and cryptographic threat models in mind.

7. Future Work

While this paper provides a security analysis of audiowmark and demonstrates practical attacks against the scheme, several directions remain open for future investigation. The most direct extension of the work is to implement the proposed countermeasures and measure their impact on detection accuracy, robustness, and computational cost. Since audiowmark has served as a basis for other watermarking schemes, it is worth investigating whether the identified vulnerabilities carry over, potentially representing broader vulnerability issues. Finally, testing whether our attacks remain effective after common signal processing operations such as MP3 compression, pitch shifting, or time stretching would provide a more complete picture of their practical robustness.

8. Author Contributions

All authors contributed to the investigation of the audiowmark repository, the identification of vulnerabilities, and the proposal of defenses. Angela and Karen were responsible for the analysis and execution of the watermark overlay, override, and forgery attack while Selinna and Jolin focused on the watermark removal attack.

9. Acknowledgments

We would like to thank professor Srini Devadas for his engaging and high quality lectures throughout the semester. We are also grateful to our TAs, Kevin He, Xiaochen Zhu, Simon Langowski, and Jophy Ye, for their guidance and support over the course of our project.

References

[1] Guangyu Chen, Yu Wu, Shujie Liu, Tao Liu, Xiaoyong Du, and Furu Wei. Wavmark: Watermarking for audio generation. 2024.

[2] Dr. N. Kavitha, Rashmika S J, and Reshika A S. Percpetual hash techniques for audio copyright protection in decentralized systems. 2025.

[3] 6.5610 Staff. Interactive proofs and zero knowledge, 2026. MIT - 6.5610 Lecture 11 Notes.

[4] Mohammad Shorif Uddin, Ohidujjaman, Mahmudul Hasan, and Tetsuya Shimamura. Audio watermarking: A comprehensive review. *International Journal on Advances in Intelligent Systems*, 2022.

[5] Stefan Westerfeld. audiowmark - audio watermarking. https://github.com/swesterfeld/audiowmark.

[6] Stefan Westerfeld. audiowmark project page. https://uplex.de/audiowmark/.