SafeSpeaker: Voice Obfuscation for Resource-Constrained IoT Devices

CAMERON HAIRE, University of Michigan, USA
YASHA IRAVANTCHI, Stanford University, USA
KANG G. SHIN, University of Michigan, USA
ALANSON P. SAMPLE, University of Michigan, USA

Embodied voice assistants offer an intuitive and powerful interaction modality, but they also pose significant privacy-leakage risks when uploading raw voice recordings to the cloud for processing. These voice recordings contain a wealth of private information beyond just spoken content, such as acoustic features that can be used to identify and track users. Existing privacy-protection approaches either rely on encryption and cloud-side processing (leaving raw audio exposed in transit and storage) or use voice obfuscation algorithms whose compute and memory demands exceed what microcontroller-class IoT devices can sustain in real time. As a result, the most resource-constrained microphone-enabled devices, which are also among the most numerous, lack any practical voice privacy protection. We present *SafeSpeaker*, a lightweight voice obfuscation framework that closes this gap by running entirely on the edge device that captures the audio, breaking the link between user identity and speech before any recording leaves the device. *SafeSpeaker* uses a time-domain, formant-approximate transformation that avoids the frequency-domain transforms used by prior DSP-based approaches, processing audio nearly 2× faster while matching the identity-obfuscation performance (equal error rate) of state-of-the-art resource-aware methods. We demonstrate the practicality of this design with a real-time prototype on a $3 Arm Cortex-M4 microcontroller that sits between a microphone and an off-the-shelf smart speaker while maintaining the usability to understand and respond to voice commands. By making voice obfuscation feasible at this hardware tier, *SafeSpeaker* extends on-device voice privacy protections to a substantially larger class of microphone-enabled smart devices.

CCS Concepts: • **Security and privacy → Privacy protections**.

Additional Key Words and Phrases: Voice Obfuscation, Voice Privacy, Resource-Aware Algorithms, IoT, Devices

ACM Reference Format:
Cameron Haire, Yasha Iravantchi, Kang G. Shin, and Alanson P. Sample. 2026. SafeSpeaker: Voice Obfuscation for Resource-Constrained IoT Devices. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 10, 3, Article 103 (September 2026), 33 pages. https://doi.org/10.1145/3831661

1 Introduction

Voice interfaces provide an accessible and intuitive way for users to interact with a variety of devices, from Internet-connected "smart speakers" to vehicle navigation systems. Voice interfaces allow convenient, hands-free interaction with these systems, enabling users to control their devices from a distance or while performing another task. The number of devices that use a voice interface is increasing. For example, more than 100 million Americans own a smart speaker [35], and voice interfaces are integrated into thermostats [13], smart refrigerators [45], and other home IoT devices. These devices use an always-listening microphone to record sound in their environment and save these recordings in a cloud server upon hearing a certain keyword (e.g. "Alexa" or "Okay Google")

Authors' Contact Information: Cameron Haire, University of Michigan, Ann Arbor, Michigan, USA, chaire@umich.edu; Yasha Iravantchi, Stanford University, Stanford, California, USA, yasha@stanford.edu; Kang G. Shin, University of Michigan, Ann Arbor, Michigan, USA, kgshin@umich.edu; Alanson P. Sample, University of Michigan, Ann Arbor, Michigan, USA, apsample@umich.edu.

ACM 2474-9567/2026/9-ART103
https://doi.org/10.1145/3831661

[4, 17, 32]. Given that these devices are commonly found in private spaces, such as the home, this operational model creates large privacy risks. New recordings may be intercepted by an adversary during transmission, and saved recordings can be revealed in a data breach [44], accessed by manufacturer employees [11, 20], or given to law enforcement without requiring a warrant [10, 21]. These recordings can contain confidential conversations and sensitive information [20], reveal the emotional state and psychological conditions of the user [12, 25], and be used to uniquely identify the user [3, 5, 18]. Furthermore, the recordings can be used to artificially clone the voice of a user, enabling impersonation-based attacks [15]. The lack of privacy protections provided to voice interfaces causes consumers to perceive a trade-off between privacy and convenience [1, 24] that discourages the use of voice interfaces.

Existing research has explored voice obfuscation—modifying the captured audio to disguise the speaker's identity before it is uploaded or stored—as a means of breaking the link between a user and their recordings. While effective, prior obfuscation approaches were designed for server- or smartphone-class hardware and assume compute and memory budgets that exceed those of microcontroller-class IoT devices by orders of magnitude. State-of-the-art ML-based obfuscation pipelines rely on feature-extraction and synthesis models that are tens to hundreds of megabytes in size [53], far beyond the tens of kilobytes of RAM and hundreds of kilobytes of flash typical of MCU-class devices. Even the lightest DSP-based obfuscation approaches rely on frequency-domain transforms, source-filter analysis, or codec-level manipulation that, in our measurements, do not run in real time on commodity microcontrollers. As a result, the most numerous and most deeply embedded class of microphone-enabled devices—smart TV remotes, voice-enabled appliances, wearables—currently has *no* practical on-device voice privacy protection available. Closing this gap requires an obfuscation algorithm whose compute and memory footprint is small enough to run in real time on commodity microcontrollers while still providing privacy protection competitive with existing techniques.

To meet this need, we present SafeSpeaker, a resource-aware voice obfuscation architecture capable of running in real time on microcontroller-class hardware. SafeSpeaker uses time-domain, formant-focused frequency modification to anonymize captured audio. Formants are an important acoustic feature for speaker identification, and prior work has explored how modifying these features affects the accuracy of automated speaker identification systems [40, 41, 62]. Whereas existing formant-based obfuscation approaches rely on frequency-domain processing, SafeSpeaker modifies formant positions entirely in the time domain using a bank of precomputed band-pass filters and per-band circular-buffer time-scale modification, eliminating the FFTs and source-filter analyses that dominate the runtime cost of prior DSP-based approaches. This design yields a nearly 2× speedup over the fastest prior DSP baselines—enough to bring real-time obfuscation within the budget of an MCU running at 170 MHz with 32 KB of RAM—while preserving competitive obfuscation performance. Because these efficiency-driven approximations reduce the intelligibility and naturalness of the modified speech, SafeSpeaker is designed for downstream voice *interfaces* rather than for human listeners; perfect naturalness is not required, only that automated speech recognition and commercial voice assistants can still understand the obfuscated commands. We verify this assumption with both ASR benchmarks and an end-to-end test against an off-the-shelf smart speaker.

Our threat model considers adversaries who gain access to stored recordings—through data breaches, insider access, or compelled disclosure—and attempt to link those recordings to a speaker's identity using automated speaker verification. We evaluate SafeSpeaker against both naive attackers, who treat obfuscated audio as if it were clean, and informed attackers, who know the obfuscation procedure and its parameters and can apply the same processing to their reference samples. We measure privacy through the equal error rate (EER) of an automatic speaker verification (ASV) system under both attack settings, and we measure utility through both the word error rate (WER) of an automatic speech recognition (ASR) model and the success rate of obfuscated commands issued to a commercial smart speaker. To characterize efficiency, we compare the per-utterance

obfuscation time of SafeSpeaker against three established DSP-based baselines, and we demonstrate end-to-end real-time operation on a commercially available microcontroller costing less than $3 USD.

Enabling real-time voice obfuscation on compute-limited devices ensures that a user's raw voice cannot be intercepted or saved in the first place. As IoT voice interfaces are integrated into more computationally demanding tasks, such as generative AI [19], on-device voice obfuscation will protect user privacy without sacrificing the convenience and functionality of voice interfaces.

The main contributions of SafeSpeaker are:

(1) A time-domain, formant-focused voice obfuscation algorithm that achieves privacy protection competitive with state-of-the-art DSP-based approaches while processing audio nearly 2× faster, bringing real-time obfuscation within reach of microcontroller-class hardware.
(2) An evolutionary parameter-optimization framework that jointly tunes band boundaries and per-band resampling coefficients against a privacy–utility fitness function combining naive EER, informed EER, and ASR WER.
(3) A complete hardware prototype on a $3 Arm Cortex-M4 microcontroller, integrated with an off-the-shelf smart speaker, demonstrating that on-device voice privacy protection is feasible at the hardware tier where the majority of microphone-enabled IoT devices reside.

2 Threat Model

Audio captured by voice interfaces is routinely uploaded to cloud databases and used by device manufacturers to improve their products and services [11, 32]. Once stored, these recordings have repeatedly been exposed to parties beyond the user: manufacturer employees who review them as part of quality-assurance pipelines [11, 20], external attackers who breach corporate databases [44], and law enforcement officers who request data without a warrant [10, 21]. We consider an adversary that has obtained access to such a database, e.g., a malicious manufacturer employee or a hacker, and aims to recover speaker identity from its contents.

We assume that recordings in the database are not labeled with user identity, but that the adversary attempts to attach such labels by exploiting the vocal characteristics that uniquely identify a speaker. A manufacturer employee may recognize the voice of a public figure, or a hacker may use machine-learning-based automatic speaker verification (ASV) to cluster recordings by speaker. Once a recording is attributed to a user, the adversary can build a profile of the user's interests and relationships, target advertisements or products, clone the user's voice for impersonation-based attacks, or exploit private information disclosed in the recording. Recordings are at risk the moment raw audio leaves the trusted capture path, so privacy protection must be applied at or near the point of capture, before the audio is exposed to any component outside the user's trust boundary. Our defense is to disguise the speaker's voice within this trusted boundary, breaking the link between identity and speech before the recording is uploaded.

We consider three tiers of adversary capability, in increasing order of strength, and design SafeSpeaker to provide protection across all three. A *naive* adversary is unaware that obfuscation has been applied and queries an off-the-shelf ASV model with the obfuscated recording against clean reference utterances. An *informed* adversary knows that obfuscation is in use and that a fixed set of obfuscation parameters has been applied; this adversary can apply the same obfuscation to its enrollment utterances before querying the ASV model, neutralizing any spectral mismatch between obfuscated and clean speech. An *adaptive* adversary additionally has access to a corpus of obfuscated speech sufficient to retrain the ASV model end-to-end on obfuscated audio, which prior voice-privacy work treats as the strongest practical attacker [51, 54]. The naive, informed, and adaptive tiers correspond to the $\text{EER}_{\text{naive}}$, EER_{inf}, and $\text{EER}_{\text{adaptive}}$ metrics reported in Section 8. The adaptive tier motivates SafeSpeaker-Rand (Section 8.2.3), a variant that rotates among multiple optimized parameter sets at the utterance granularity to deny the adaptive adversary a stationary obfuscation function to learn against.

Several attack vectors are out of scope for this work. We do not consider attacks that require physical access to the recording device, since such access already compromises the audio capture path before any obfuscation can be applied. We do not prevent an adversary from cross-referencing recordings with private information—names, addresses, account numbers—that the user explicitly speaks aloud, since this is a content-leakage problem orthogonal to identity obfuscation. Likewise, we do not address the leakage of speaker *attributes* such as gender, accent, or apparent age; SafeSpeaker is designed to defeat *identity* re-identification (matching a recording to a specific known speaker), and prior work has shown that hiding coarse demographic attributes typically requires more aggressive content-altering transformations that are incompatible with maintaining ASR utility on a voice interface [55]. Finally, we do not aim to prevent the voice interface from recording audio during an inadvertent activation, but the audio captured in such a case is still protected by the on-device obfuscation pipeline.

3 Related Work

Voice interfaces have prompted two complementary lines of research: studies of how users perceive privacy risks around voice assistants, and systems that mitigate those risks technically. User studies consistently report that consumers are uncomfortable with how these devices record and store audio, and are often unaware of the extent of that collection [7, 27, 63]. These concerns are reinforced by reports of audio leaks to manufacturer employees [20], unintentional recording of private conversations [22], and recorded audio being used for ad targeting [56]. They have also prompted regulatory responses, including the General Data Protection Regulation (GDPR) [59], the California Privacy Rights Act [36], and NIST's IoT consumer-labeling guidance [37], all of which constrain how consumer audio can be collected and reused. The technical line of work, which is the focus of this section, splits into two threads: systems that prevent the recording of extraneous audio in the first place, and systems that obfuscate speaker identity within recordings that are made.

3.1 Audio Recording Protections

A first thread of prior work focuses on preventing voice interfaces from recording audio that the user did not intend to capture, such as ambient conversations or background activity. Several proposals enforce stricter capture conditions on smart speakers: Mhaidli *et al.* [31] require the user to look at the device before the microphone is enabled, Sun *et al.* [49] use targeted acoustic jamming to suppress recording, and Chandrasekaran *et al.* [7] propose a remote power-cut mechanism that disables the device entirely. These approaches reduce leakage from unintentional recordings, but they offer no protection for the recordings that the user does intend to make. SafeSpeaker is complementary: it leaves the capture decision to the user and the device, and instead reduces the speaker-identifying information that any captured recording carries.

3.2 Voice Obfuscation

A second thread of prior work, and the one most directly related to SafeSpeaker, modifies the captured audio itself to disguise the speaker's identity before the recording is uploaded. Voice-obfuscation systems broadly fall into two categories: machine-learning (ML) approaches that synthesize obfuscated audio from learned representations, and digital signal processing (DSP) approaches that transform the waveform directly [6]. ML-based methods generally achieve strong privacy while preserving the linguistic content needed for downstream voice applications, but their compute and memory footprints exceed the capacity of microcontroller-class IoT devices by orders of magnitude. DSP-based methods are substantially lighter and have therefore been the dominant choice for resource-aware deployments, though as we discuss below, the lightest existing DSP pipelines still rely on frequency-domain analysis that is too expensive for real-time operation on MCU-class hardware.

3.2.1 Machine-Learning Based Obfuscation. ML-based obfuscation is widely used because it can produce intelligible, natural-sounding speech that disguises the speaker's identity. Most ML approaches fall into three groups,

none of which is currently realizable on microcontroller-class hardware. The first group decouples linguistic content from speaker identity and re-synthesizes the utterance with an anonymized speaker representation, typically an x-vector. Fang *et al.* [14] construct anonymized x-vectors by averaging vectors drawn from a pool of other speakers, and follow-up work has explored alternative sampling strategies, including drawing from dense regions of the x-vector space [47] and synthesizing artificial x-vectors with a generative network [30]. These pipelines preserve usability well, but their compute and memory footprints place them out of reach for embedded deployment: the linguistic and x-vector feature extractors published by the Voice Privacy Challenge [52] are 84 MB and 32 MB respectively, while typical microcontrollers offer tens of kilobytes of RAM and hundreds of kilobytes of non-volatile storage, a gap of more than two orders of magnitude. The second group adds an adversarial perturbation to the recording so that an ASV model misclassifies the speaker. Zhang *et al.* [64] optimize a static additive perturbation against a known ASV model, and V-Cloak [8] uses a generative network to produce input-conditioned perturbations. The static-perturbation construction is light enough to run on constrained hardware, but it assumes white-box access to the target ASV model and was not evaluated against an adaptive attacker that retrains on perturbed audio; the generative-perturbation construction restores robustness but reintroduces the model-size problem of the first group. The third group sidesteps audio upload entirely by transcribing speech to text on the device and uploading only the transcript. MegaMind [50] routes voice-assistant requests through a local speech-to-text pipeline, and Meyer *et al.* [29] pair speech recognition with text-to-speech to re-render anonymized audio. Both approaches assume an on-device ASR model, and current ASR models small enough to run on an MCU sacrifice the recognition accuracy required for a useful voice interface. Across all three groups, the limiting factor for IoT deployment is the same: the feature-extraction or synthesis models that make ML-based obfuscation effective also make it incompatible with the compute and memory budgets of microcontroller-class hardware.

3.2.2 Digital Signal Processing Voice Obfuscation. DSP-based voice obfuscation aims to deliver speaker-identity protection at a small fraction of the compute and memory cost of ML pipelines, making it the natural starting point for resource-constrained voice interfaces such as smart speakers, wearables, and TV remotes. Like their ML counterparts, DSP methods modify the acoustic features that an ASV system relies on, principally fundamental frequency, tempo, and the spectral envelope. Existing DSP methods divide into two sub-families. The first applies a frequency-warping function to the speech signal to redistribute its frequency content. VoiceMask [41] segments an utterance, estimates each segment's fundamental frequency, and warps the segment's spectrum with a tunable warping function; Mawalim *et al.* [28] take a related resample-and-overlap-add approach that achieves a similar warping effect in the time domain. The second sub-family targets formant locations directly, since formants encode much of a speaker's vocal-tract identity. The McAdams transformation, popularized in the voice-privacy literature by Patino *et al.* [40], perturbs the poles of an LPC vocal-tract model to shift formant peaks, while MicPro [60] reaches the same objective by manipulating the line-spectral-frequency representation used inside the CELP codec.

Although these DSP methods are far lighter than ML pipelines, none of them have been shown to run in real time on commodity microcontrollers: each still depends on either an FFT-based spectral analysis [28, 41], an LPC analysis-resynthesis loop (McAdams), or codec-internal manipulation (MicPro). SafeSpeaker is also a formant-modification approach, but it replaces the per-frame source-filter analysis used by McAdams and MicPro with a precomputed band-pass filterbank and per-band time-scale modification, eliminating the dominant runtime cost of those baselines. As we show in Section 8, this design lets SafeSpeaker match the privacy protection of the strongest DSP baselines under naive, informed, and adaptive attackers while reducing obfuscation latency by a factor of two and shrinking the memory footprint by an order of magnitude relative to existing DSP pipelines. Together, these reductions move real-time voice obfuscation from smartphone- or smart-speaker-class processors

down to $3 microcontroller-class hardware, extending on-device privacy protection to the most numerous and most resource-constrained tier of microphone-enabled IoT devices.

4 Obfuscation Background

This section provides the acoustic background needed to understand SafeSpeaker's obfuscation strategy. We first describe the speaker-identifying features that SafeSpeaker targets, then position our approach against the formant-modification techniques used by current state-of-the-art DSP-based obfuscation methods.

4.1 Fundamental Frequency and Formants

Fundamental frequency and formant positions are central to both the linguistic interpretation of speech and the identification of the speaker who produced it [2, 23]. The standard source-filter model treats speech production as an excitation generated by the larynx that is shaped by a filter formed by the vocal tract [16]. The fundamental frequency is set by the larynx excitation, while the formant frequencies are the resonant peaks of the vocal-tract filter. Figure 1 illustrates this in the vowel /æ/ (the "a" in "had") spoken by an adult male: the peaks of the spectral envelope mark the formant locations. Because both the fundamental frequency and the formant frequencies depend on the geometry of an individual's vocal tract, they encode information that is characteristic of that speaker. Modifying these features alters the perceived identity of the recorded speaker, breaking the link between the recording and the original speaker's voice.

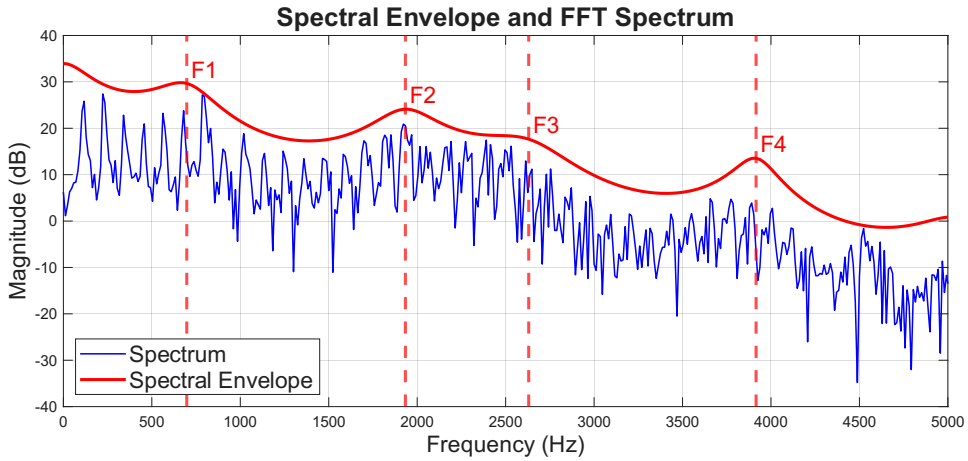


Fig. 1. The spectrum (blue) and spectral envelope (red) of /æ/ as spoken by a male speaker. The peaks in the spectral envelope, labeled with dashed vertical lines, correspond to the locations of the formant frequencies.

4.2 Formant Modification Techniques

Several of the DSP-based approaches introduced in Section 3.2.2 involve formant modification as part of the voice obfuscation process. Specifically, VoiceMask [41], the McAdams transformation approach [40], and MicPro [60] all produce audio with modified formant locations. The following will briefly introduce each method, give an overview of the modification approach presented in this work, and discuss how our method modifies speaker-identifying acoustic features through formant modification by comparing the spectral envelope of audio modified using SafeSpeaker with spectral envelopes of audio modified using existing DSP-based voice obfuscation.

4.2.1 VoiceMask. VoiceMask [41] adapts vocal-tract-length normalization (VTLN) for privacy. VTLN was originally developed to reduce inter-speaker variability by warping the frequency content of an utterance to compensate for differences in vocal-tract length, since vocal-tract geometry directly determines formant locations. VoiceMask repurposes this idea: rather than normalizing speakers toward a common reference, it applies a compound frequency-warping function whose effect is to push formant peaks away from their original positions, breaking the mapping between the recorded formants and the speaker's vocal-tract geometry. Figure 2 illustrates VoiceMask's effect on the spectral envelope.

4.2.2 The McAdams Transformation. The McAdams transformation [40] leverages source-filter analysis via linear predictive coding (LPC) to identify and modify formant locations in recorded speech. LPC models the vocal tract as a filter and finds the coefficients of this filter, which correspond to the formant locations. The authors used these coefficients to derive pole angles for the vocal-tract filter model. Modifying the poles results in a change in the distribution of the frequency content, altering the formant locations and the timbre of the original speech. One key difference between this approach and the approach used in VoiceMask [41] is that the LPC process excludes the location of the fundamental frequency, meaning that the underlying pitch of the speech is not altered by the McAdams transformation. Figure 2 shows the effect that the McAdams transformation has on the formant locations.

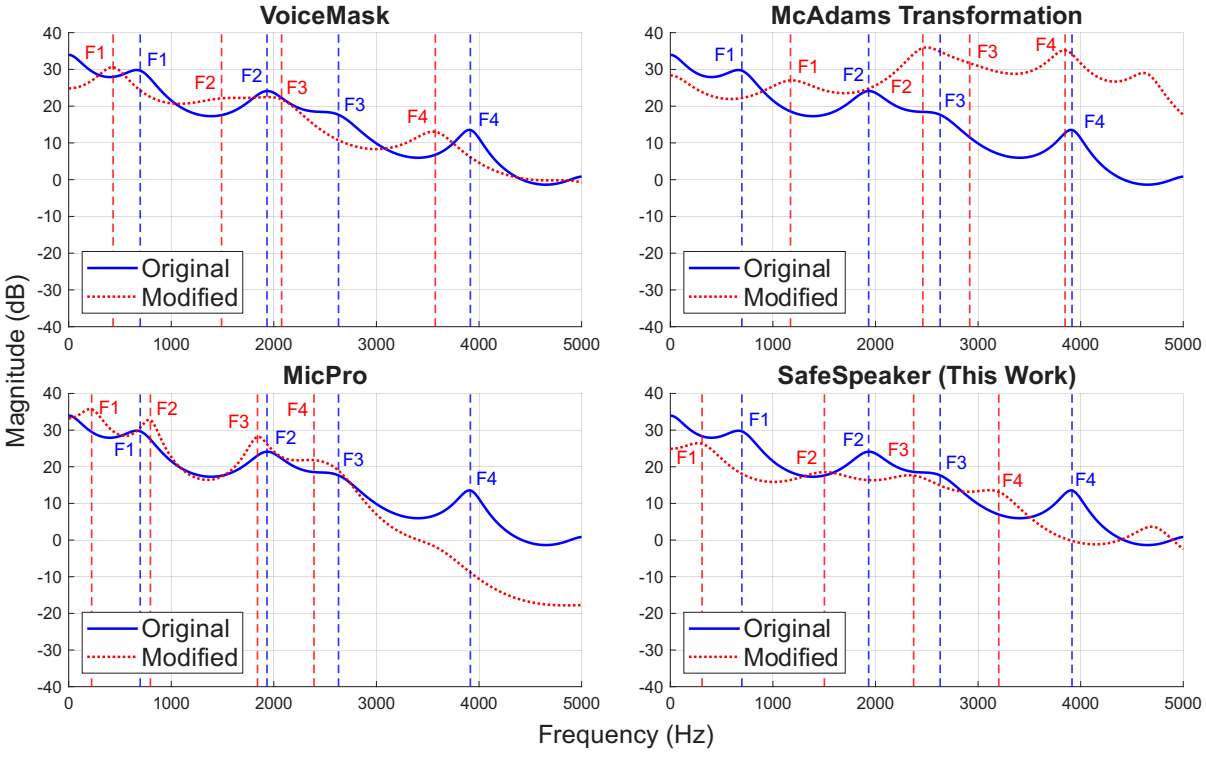


Fig. 2. The spectral envelopes /æ/ as spoken by a male speaker after being modified by various voice obfuscation approaches. The shifts in the locations of the peaks (shown with vertical dashed lines) correspond to the change in the formant locations. 2(d) shows how the method proposed in this work is able to successfully change the formant locations in order to protect the speaker's privacy.

4.2.3 MicPro. Similarly to the McAdams transformation [40], MicPro [60] uses a formant modification approach based on LPC, but leverages how LPC is used in the CELP audio codec to encode audio. Rather than using LPC coefficients, the CELP audio codec uses line spectrum frequencies (LSFs), which are derived from the LPC analysis. Although LPC coefficients and LSFs represent the same information in different ways, LSFs are more robust against transmission errors, which makes them more appealing for audio encoding. The authors of MicPro modify the CELP codec directly such that encoded audio is obfuscated by altering formant locations. Figure 2 shows the effect that MicPro has on the formant locations.

4.2.4 SafeSpeaker (This Work). Whereas the three DSP methods above all rely on a per-utterance source-filter or codec-level analysis to locate formants before modifying them, SafeSpeaker performs no such analysis at runtime. Instead, the input signal is split into a small number of fixed frequency bands by a precomputed band-pass filterbank, and each band is independently resampled to shift its frequency content. The band boundaries and per-band resampling coefficients are chosen once, offline, by an evolutionary optimization procedure (Section 7) that targets speaker-identifying frequency locations across speakers. Critically, this optimization is performed once per device population rather than once per user: a single parameter set is produced ahead of deployment using a held-out development corpus, then shipped with the device firmware. No optimization, training, or model inference takes place on the device or per-user, so the runtime cost on the IoT endpoint is exactly the cost of the filterbank and resampling operations described above. We show in Section 8 that a parameter set tuned in this way generalizes to unseen speakers, datasets, and noise conditions. The same offline procedure can also produce multiple parameter sets that the device rotates among at runtime, a variant we develop in Section 8.2.3 (SafeSpeaker-Rand) to defend against adaptive attackers that retrain on obfuscated speech. This factoring (offline parameter search, runtime filterbank-plus-resampling) replaces every FFT, LPC, and codec operation that the baselines rely on, and is the source of SafeSpeaker's reduction in latency and memory footprint relative to those baselines. Despite avoiding any explicit formant tracking, the resulting transformation perturbs the spectral envelope in qualitatively the same way as VoiceMask, the McAdams transformation, and MicPro, as Figure 2 illustrates. Sections 5, 6, and 7 present the full implementation.

5 System Overview

This work presents a lightweight, real-time voice obfuscation algorithm designed for resource-constrained systems such as microcontroller-class devices. This section will provide a high-level overview of the construction of the proposed obfuscation system. Section 6 (Audio Obfuscation) and Section 7 (Obfuscation Parameter Optimization) discuss the details of each component in more depth.

The approach presented in this work centers around splitting incoming audio into N predetermined frequency regions. Each region is resampled by a different factor to change the frequency content and the output audio is constructed by combining the modified regions.

The left half of Figure 3 shows how the frequency regions and resampling coefficients are determined. To provide a strong level of privacy protection, an evolutionary algorithm is used to explore the parameter search space and find the best tradeoff between privacy and usability. Initially, the frequency regions are informed by the average locations of speech formants. The evolutionary algorithm adjusts the size, position, and resampling coefficients of these regions to provide a strong level of privacy protection.

The right half of Figure 3 shows an overview of the audio processing pipeline. The input signal is divided into frequency regions determined by the evolutionary algorithm using a bank of infinite-impulse response (IIR) filters. The frequency content of each region is modified by time-scale modification (TSM) using a circular buffer. The modified regions are summed together to form the obfuscated audio.

Our approach achieves greater computational efficiency compared to state-of-the-art DSP-based approaches by using precomputed frequency bands to estimate the locations of speech formants, an acoustic feature that is

important for speaker identification tasks. Compared to the source-filter analysis and bidirectional frequency domain transforms used in prior work, using estimated formant locations introduces a significant decrease in the resources required to perform voice obfuscation in real time.

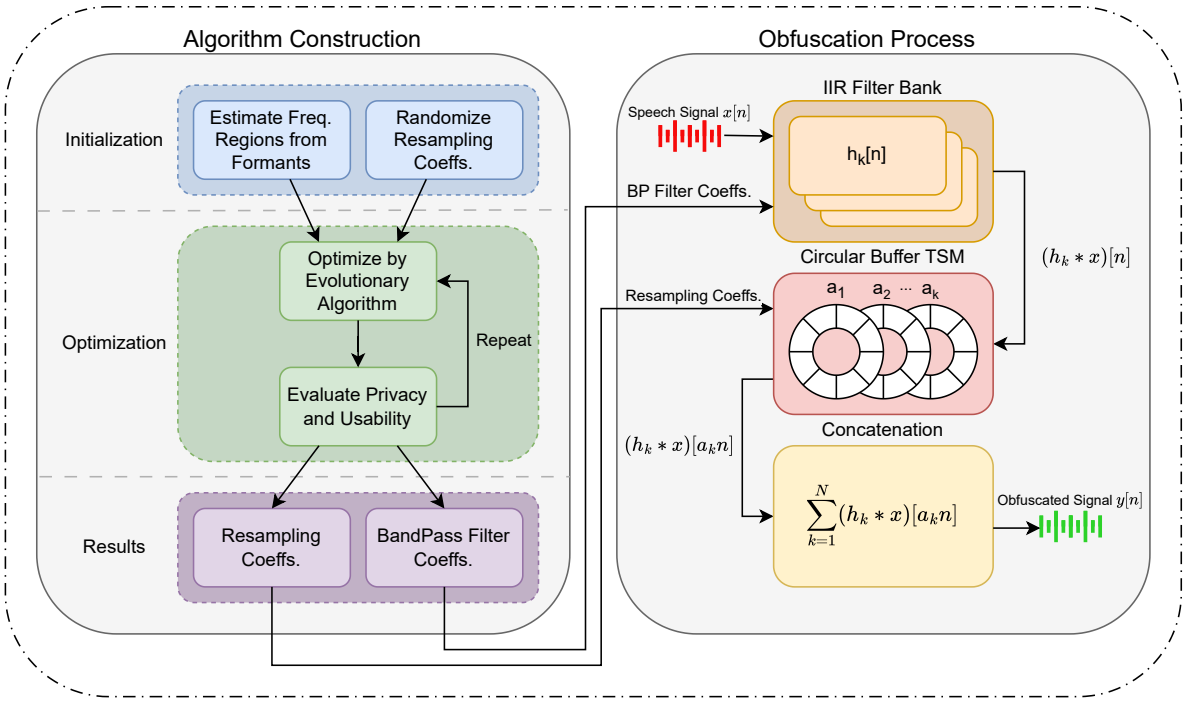


Fig. 3. Block diagram of the proposed obfuscation strategy. Captured audio is isolated into multiple frequency regions by a band-pass filterbank. Each region is resampled to modify the frequency content and the modified regions are combined to produce the obfuscated audio.

6 Audio Obfuscation

Section 5 introduced the obfuscation algorithm at the block-diagram level of Figure 3. This section presents the algorithm in full detail, covering the parameter definitions, filter and buffer designs, and signal-representation choices needed to reproduce our implementation. The audio-modification pipeline is engineered to fit within a microcontroller-class compute envelope: MCUs used in voice-IoT products typically offer processor frequencies in the tens-to-hundreds of megahertz, tens of kilobytes of RAM, and hundreds of kilobytes of flash. The ARM Cortex M4 (part number: STM32G431KB) used in the prototype of Section 9, for example, runs at up to 170 MHz with 32 KB of RAM and 128 KB of flash, and costs under $3 in volume. Combined with on-chip audio peripherals, this resource envelope makes such MCUs an attractive choice for voice-enabled embedded systems like smart thermostats, but it also rules out the FFT- and LPC-based pipelines used by prior DSP obfuscation methods and shapes the design choices in the rest of this section. Section 6.1 defines the parameters that govern the algorithm's behavior, and Section 6.2 describes the runtime pipeline that applies them.

6.1 Obfuscation Parameters

As introduced in Section 5, the algorithm divides incoming speech into N frequency bands via a bank of precomputed IIR filters, resamples each band by a per-band coefficient a_k, and sums the resampled bands to form the output. Obfuscating audio according to our proposed technique requires choosing values for the following three parameters:

- The number of frequency regions to modify: N
- The lower bound and the upper bound of each frequency region: $f_{L,k}$ and $f_{H,k}$, respectively
- The resampling coefficient for each frequency region: a_k

The choice of these values determines the algorithm's privacy strength, audio usability, and compute and memory cost; we discuss how each parameter is set in the sub-subsections below, and how their values are jointly optimized in Section 7. Figure 4 provides a visual description of the obfuscation parameters on the spectral envelope of the vowel /æ/. Formant locations are overlaid to show how the bands align with the speaker-identifying acoustic features the algorithm targets.

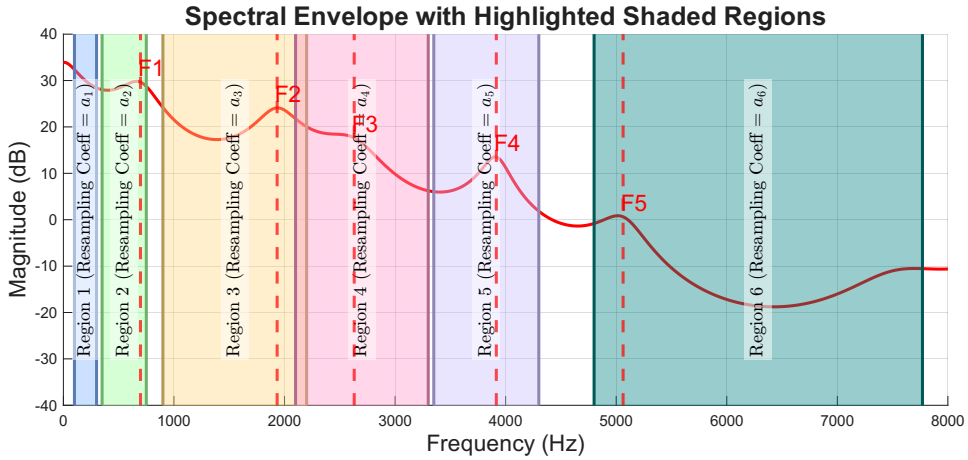


Fig. 4. An example of the positions and sizes of the frequency regions modified by the proposed obfuscation algorithm. The regions are randomly initialized to ensure sufficient coverage of the search space, then optimized through an evolutionary algorithm. The spectral envelope of /æ/ is included to show the relationship between the frequency regions and the formant positions (marked with vertical dashed lines.)

6.1.1 Frequency Region Count. The frequency region count, N, represents the number of frequency regions that are modified as part of the obfuscation process. As discussed in Section 4, fundamental frequency and formant positions are important acoustic features for determining speaker identity, and SafeSpeaker protects an individual's privacy by altering these acoustic features. The higher the number of modified frequency regions, the higher the degree of control the obfuscation algorithm has over the output audio. However, introducing additional modified frequency regions adds computational complexity. Section 7.3.1 details how we choose an appropriate value of N.

6.1.2 Frequency Region Location. The sizes and positions of each of the N modified frequency regions are learned through an evolutionary optimization process discussed in Section 7. During the optimization, the band boundary values are chosen at random between 1 and 7999 Hz to both ensure ample coverage of the search space and

maintain compatibility with the band-pass filterbank. Furthermore, these parameters are learned offline, reducing the amount of computation required to obfuscate audio. The optimization process is responsible for learning which combinations of frequency regions produce the greatest privacy protections without compromising the usability of the obfuscated audio. The colored regions in Figure 4 serve as a visualization of the frequency region locations.

6.1.3 Resampling Coefficients. Each filtered band is resampled by factor a_k to modify the frequency content. The value of the resampling coefficient determines whether the frequency content in a band is shifted higher or lower. For example, if $a_k = 0.5$, then $Y(f) = X(\frac{f}{0.5})$. Similarly to the frequency region locations, the resampling coefficient applied to each frequency region is learned through the optimization process discussed in Section 7. The initial value of a_k is uniformly randomly sampled such that $0.1 \leq a_k \leq 1.9$. These values were chosen based on initial observations of how the resampling coefficient impacts audio intelligibility. When shifting an entire utterance by a value of $a_k < 0.1$, the obfuscated audio is completely unintelligible; similarly, when shifting an utterance by a value of $a_k > 1.9$, the obfuscated audio is also largely unintelligible. However, during the optimization process, a_k is allowed to take on values that are outside of this range, as we observed that such resampling coefficients can increase the overall obfuscation effect when applied to small frequency ranges. Figure 4 shows how each frequency region is assigned a different resampling coefficient.

6.2 Obfuscation Process

Our audio processing pipeline is divided into three steps: frequency band extraction using a bank of N IIR filters, frequency modification by time-scale modification (TSM) using a set of N circular buffers, and the concatenation of all N modified frequency bands. In order to reduce memory requirements and allow for real-time audio streaming, our audio modification operates on small segments of audio and performs all frequency modifications in the time-domain.

6.2.1 Band-pass Filtering. The first step in the audio modification process is to split an incoming utterance into N frequency bands using a set of N fourth-order IIR Butterworth band-pass filters. Figure 4 shows a set of example frequency bands superimposed on the spectral envelope of /æ/. IIR filters were chosen for their relative computational simplicity when compared to alternate options such as FIR or FFT based filters. As mentioned in Section 6.1.1, we set $N = 6$ in our evaluation to reduce memory overhead.

The input audio segment $x[n]$ is passed through the N filters to produce N filtered segments. If the impulse response of each of the N band-pass filters is defined as $h_k[n]$ for an integer k such that $1 \leq k \leq N$, then each of the filtered audio segments can be expressed as $y_k[n] = (x * h_k)[n]$.

6.2.2 Circular Buffer Time-Scale Modification. Once the input audio has been filtered into the N frequency regions, each region can be resampled to modify the frequency content. The obfuscation algorithm proposed in this work uses a different resampling coefficient, a_k, for each region to independently alter the strength and direction of the frequency modification applied to each region. We observe that this produces a stronger level of privacy protection compared to using a single resampling coefficient for the entire spectrum.

Frequency modification is performed using a method based on circular buffers. Figure 5 provides a visual explanation of how audio is resampled using the circular buffer. At each timestep, one sample from the input, $x[n]$ is written into the buffer and one value from the buffer is written to the output, $y[n]$. The point where $x[n]$ is written and the point where $y[n]$ is read are not always the same location within the buffer. To produce the frequency-shifting effect, the two locations increment at different rates. Our obfuscation algorithm increments the write location by one buffer index and the read location by a_k. a_k is not guaranteed to be an integer, and non-integer read locations are rounded down. N circular buffers are used to simultaneously modify the N frequency regions.

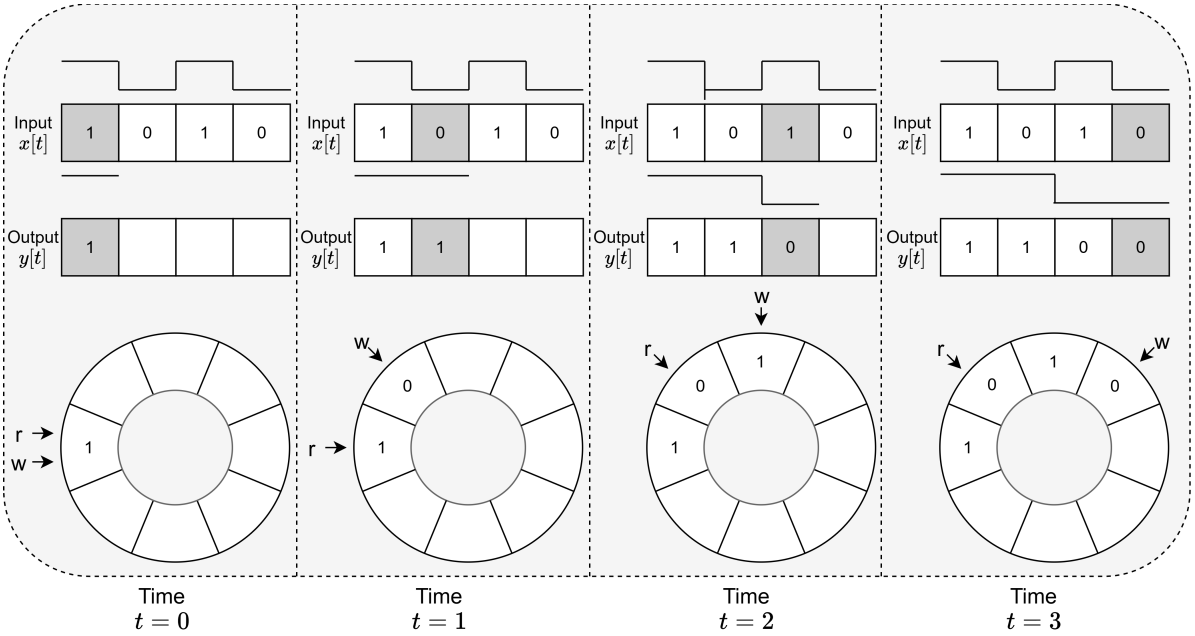


Fig. 5. A visual explanation of the process by which the circular buffer time-scale modification changes the frequency content of an input. The example displays four time-steps and a resampling coefficient of 0.5. At each timestep t, the value from the input, $x[t]$ is written into the buffer at the write location ("w") and the value at the read location ("r") is read into the output $y[t]$.

The circular construction of the buffer means that the read and write locations will inevitably intersect, as one will "chase" the other around the buffer. When this occurs, there is a perceivable discontinuity in the output audio. To remove this, the final output value read from the buffer is a weighted sum of the value at the read location and the value opposite the read location in the buffer. When the write location is about to intersect with the read location, the weights favor the value on the opposite side of the buffer to remove the discontinuity.

6.2.3 Reconstruction. The final step combines the N modified bands into a single output signal. Letting $\tilde{y}_k[n]$ denote the resampled output of band k, the obfuscated signal is the sum

$$y[n] = \sum_{k=1}^{N} \tilde{y}_k[n]. \tag{1}$$

The circular buffer resampling method ensures that the number of samples in each chunk does not change as a result of the resampling. This means that the shifted chunks do not need to be aligned before they are combined.

7 Obfuscation Parameter Optimization

The number of frequency bands, frequency-band edges, and per-band resampling coefficients introduced in Section 6 jointly determine how strongly SafeSpeaker protects speaker identity and how much that protection affects audio usability. While the number of frequency bands is fixed according to the memory budget of the target device (Section 7.3.1), the band edges and resampling coefficients are optimized according to an evolutionary algorithm (EA). The use of an EA ensures that the obfuscation uses parameters that provide a strong degree of

privacy protection while maintaining the usability of the obfuscated audio. An EA is well-suited for this task due to the nature of the performance evaluation, which relies on downstream ASV and ASR models rather than closed-form loss. The optimization runs offline on a development corpus and produces sets of obfuscation parameters that can be frozen and included in the firmware of privacy protecting voice-interfaces; the optimization does not require additional on-device computation or user-specific training data. Sections 7.1, 7.2, and 7.2.1 introduce the metrics used to evaluate the candidate solution, fitness function design, and how the fitness function is evaluated. Section 7.3 discusses the exploration of the design space.

7.1 Fitness Function Metrics

Choosing arbitrary values for the parameters introduced in Section 6.1 does not guarantee that a user's voice will be sufficiently hidden or that the obfuscated audio will be compatible with downstream voice services. The evolutionary algorithm evaluates each solution according to a fitness function to determine which solutions are the most promising. This process is repeated over multiple iterations to progressively tune the obfuscation parameters.

7.1.1 Privacy Metric. To evaluate the privacy protections offered by a set of obfuscation parameters, we leverage the equal error rate (EER). EER is a common metric used in authentication systems to express how well the system balances the false accept rate (FAR) and false reject rate (FRR). More specifically, the EER is equal to the value of the FAR or the FRR when FAR = FRR. In the context of voice obfuscation, measuring the difference in the EER when using clean audio and when using obfuscated audio shows us how well the obfuscation hides the identity of the speaker. If the obfuscation is able to disguise the speaker's identity, the authentication error will grow larger, causing the EER to increase. The ideal voice obfuscation will cause an authentication system to perform no better than chance. This occurs when EER = 0.5. At this value, both the FAR and the FRR are also 0.5; half of the utterances from other speakers are identified as the target speaker and half of the utterances from the same speaker are identified as other speakers. It should be noted that swapping the classification outputs ("target speaker" and "other speaker") has the effect of inverting the EER ($EER_{new} = 1 - EER_{old}$), which means that an EER > 0.5 implies weaker privacy protections than EER = 0.5.

The EER is determined using an automatic speaker verification (ASV) model. An ASV model finds the speaker similarity score between two utterances, $S(u_i, u_j)$. This score is compared to a threshold, θ, to determine whether the individual in the first utterance u_i, is the same as the individual in the second utterance, u_j. Utterances are considered to feature the same individual if $S(u_i, u_j) \geq \theta$ and different individuals if $S(u_i, u_j) < \theta$. The value of the EER is determined by changing the similarity threshold until FAR = FRR. Changing the value of θ changes the FAR and FRR, and the EER occurs when FAR = FRR.

To provide robust privacy protections, we consider two forms of the EER in the design of our fitness function. The first form of the EER is computed by comparing clean speech with speech obfuscated by an algorithm $O(u, p)$, where u is the utterance and p is the set of obfuscation parameters. This comparison can be expressed as $S(u_i, O(u_j, p))$ for a fixed set of parameters p. This form of the EER represents how well the speaker's identity is hidden from the ASV model. In existing work, this EER is often referred to as the "naive EER" (EER_{naive}), as it mirrors how an ASV model may be used to identify speakers if an attacker is unaware that obfuscation is in place. The second form of the EER is computed using the same similarity function used for the naive EER, but to compare two utterances obfuscated with the same set of parameters p. This comparison can be expressed as $S(O(u_i, p), O(u_j, p))$. This form of the EER represents how robust the obfuscation algorithm is against a more complex attack and is often referred to as the "informed EER" (EER_{inf}). In this scenario, the attacker is aware of the obfuscation method and the obfuscation parameters [48]. Figure 6 provides a visual explanation for how EER_{naive} and EER_{inf} are determined.

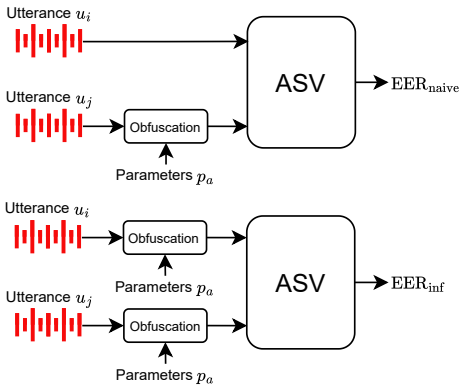


Fig. 6. $\text{EER}_{\text{naive}}$ is determined by comparing a clean utterance u_i to an utterance u_j that has been obfuscated using a set of parameters p_a. EER_{inf} is determined by comparing two utterances, u_i and u_j, that have both been obfuscated using the same set of parameters p_a. Both sets of comparisons are performed using the same automatic speaker verification (ASV) model.

7.1.2 Usability Metric. The obfuscation algorithm that we propose in this work is designed with resource-constrained systems in mind. Although our proposed algorithm is able to achieve a level of privacy protection comparable to current state-of-the-art DSP-based voice obfuscation techniques with less computational overhead, our obfuscation scheme is more likely to degrade the quality of the audio as a result of the low-overhead processing methods. To ensure that the obfuscated audio retains compatibility with downstream voice services, such as voice assistants, we include a measure of the obfuscated audio's intelligibility in our fitness function.

To evaluate the intelligibility of obfuscated speech, we measure the word error rate (WER) of each obfuscated utterance. The WER measures the proportion of words that are incorrectly transcribed by an automatic speech recognition (ASR) model. Similarly to the EER, the WER is used in the evaluation of existing DSP-based voice obfuscation systems. A lower WER indicates that downstream ASRs or transcription models will be able to correctly interpret the obfuscated speech. The goal of the evolutionary algorithm is to maintain the usability of the audio according to industry standards, which indicate that a WER less than or equal to 0.2 is appropriate for voice interfaces [57].

7.2 Fitness Function Design

We form our fitness function by first adjusting each of the two metrics discussed in Section 7.1 to encourage the exploration of solutions that result in the strongest privacy given that a sufficient level of usability has been achieved.

As mentioned in Section 7.1.1, if EER > 0.5, then it can be inverted by swapping the classification outputs. To account for this, we mirror any EER greater than 0.5 about 0.5. The EER is then multiplied by 2 to fit within [0,1], placing it on the same scale as the WER. We consider $\text{EER}_{\text{naive}}$ and EER_{inf} to be of equal importance to the parameter optimization. To reflect this, the two values are averaged in the final fitness function. The equation used to adjust both EERs is shown in Eq. 2.

$$\text{EER}_{\text{adj}} = 1 - |2 \cdot \text{EER} - 1| \tag{2}$$

Section 7.1.2 motivates maintaining a WER of at least 0.2. Thus, we want our optimization to prioritize the WER until it achieves a value of at most 0.2, then focus on increasing the EER while maintaining the WER. We start by inverting the WER to provide high fitness values for low WERs. Initial observation of the optimization process revealed that a weight of 0.25 for WER ≤ 0.2 yielded final solutions with the best balance of privacy and usability. For WER values greater than 0.2, we chose to use a linearly scaled reduction in fitness. This method has the advantage of guiding the initial population toward the target WER by providing continuous fitness gains for decreasing WER. In contrast, using a fixed negative penalty would not provide the algorithm with the information to favor solutions with a WER close to 0.2. We chose to give a fitness score of 0 for solutions with a WER that was twice the target value. To avoid a discontinuity between fitness values for WER ≤ 0.2 and WER > 0.2, the slope and intersection are chosen such that the two WER fitness curves intersect at WER = 0.2. Eq. 3 shows the piecewise function that accomplishes these objectives.

$$\mathrm{WER}_{\mathrm{adj}} = \begin{cases} 1 - .25 \cdot \mathrm{WER}, & \mathrm{WER} \leq 0.2 \\ 1.9 - 4.75 \cdot \mathrm{WER}, & \mathrm{WER} > 0.2 \end{cases} \tag{3}$$

The complete fitness function is shown in Eq. 4.

$$\frac{\mathrm{EER}_{\mathrm{naive,adj}} + \mathrm{EER}_{\mathrm{inf,adj}}}{2} + \mathrm{WER}_{\mathrm{adj}} \tag{4}$$

7.2.1 Fitness Function Evaluation. To evaluate each candidate solution according to the fitness function defined in Eq. 4, audio modified by each candidate solution is processed by ASV and ASR models provided by SpeechBrain [43]. The ASV model is SpeechBrain's `spkrec-ecapa-voxceleb`, an ECAPA-TDNN model trained on the VoxCeleb dataset [9, 33, 34], and the ASR model is SpeechBrain's `asr-wav2vec2-librispeech`, a wav2vec model trained on the LibriSpeech ASR corpus [39].

The evaluation set is designed to be small enough to allow efficient exploration of the search space while still providing information on how each candidate performs. In order to reduce the risk of overfitting to the training data, the utterances used to evaluate the candidate solutions are randomized at the start of each generation. Each generation uses five enrollment and ten trial utterances from each of the 30 of the speakers in the VCTK_dev dataset. To determine the EERs, each of the enrollment utterances is compared against every trial utterance using the ASV model. To calculate the WER, the modified utterances are transcribed by the ASR model. The resulting transcripts are compared to the ground-truth transcripts provided by the VCTK dataset.

7.3 Design Space Exploration

To investigate how the obfuscation parameters impact the privacy–usability tradeoff of modified audio, we conducted a series of evaluations centered around the parameter search space. Specifically, we investigated the effects of changing the number of modified frequency regions, ablating sets of modification parameters, and performing multiple optimization passes with different starting seeds. In what follows we will detail the results of each of these evaluations.

7.3.1 Band Count Exploration. To choose a value for the frequency region count, N, we evaluated how the privacy and usability of our obfuscation algorithm changes with the number of modified frequency regions. To ensure a fair comparison between different region counts, an optimized parameter set for each region count was produced using a shortened version of the optimization procedure discussed in Section 7. The optimization process was shortened to aid efficient exploration of the design space. In the shortened version of the optimization, evaluating one candidate parameter set consisted of 5400 pairwise comparisons and 150 transcriptions. Each generation consisted of 150 candidates and each optimization ran for 200 generations. At the end of each optimization, the

fitness of the best parameter set was recorded for comparison. The fitness, defined in Eq. (4), encapsulates both the privacy and the usability provided by a parameter set into a single value.

In addition to comparing different region counts according to their fitness, we also measured how the number of modified frequency regions impacts the memory and compute overheads associated with obfuscation. To evaluate these metrics, we implemented our algorithm on an STM32G431KB microcontroller, the same microcontroller used to develop the hardware instantiation discussed in Section 9. Memory overhead was calculated by adding static RAM usage taken from the compiled `.elf` binary to the stack usage reported by output `.su` files. Compute overhead was evaluated using the real-time factor, which is defined as the ratio of the obfuscation time to the length of the audio being obfuscated.

Figure 7 shows the results of our investigation. Larger numbers of modified frequency regions achieve higher fitness at the cost of increased memory and compute overhead. The memory capacity of the STM32G431KB and other off-the-shelf microcontrollers are shown through dashed lines on the right side of Figure 7. The memory and compute overhead of region counts that exceed the memory capacity of the STM32G431KB used in our evaluation were estimated based on the previous values, rather than from direct measurement. Given the constraints of the devices used in this work, we chose $N = 6$, which provides the best performance for a given memory capacity.

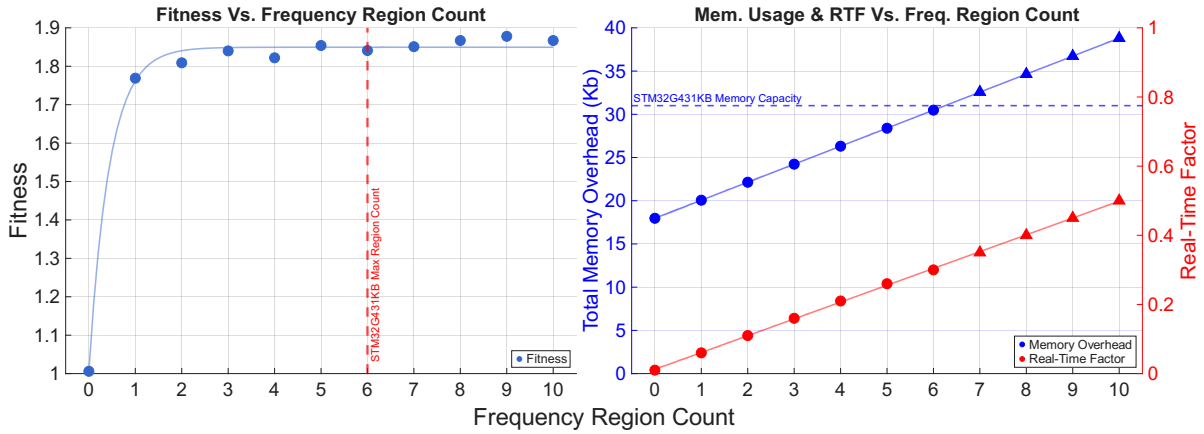
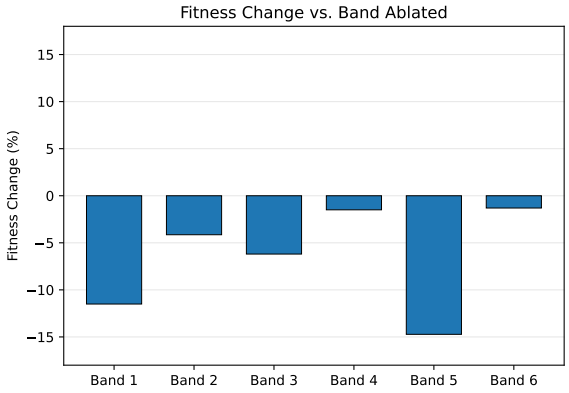


Fig. 7. (Left) The EER and WER values achieved by differing numbers of frequency regions after optimization using a shortened evolution. (Right) The real-time ratio and RAM usage for differing number of frequency regions. RTF was measured using an STM32G431KB. Red horizontal lines show the RAM capacities of various low-cost, off-the-shelf microcontrollers.

7.3.2 Band Ablation. The optimization process learns the combination of frequency regions and resampling coefficients that achieves the highest fitness score. To investigate how each frequency region and its corresponding resampling coefficient impact the fitness achieved by the overall solution, we performed an ablation study. To remove the effect that modifying a given frequency region has on the overall obfuscation performance, we set the target frequency region's resampling coefficient equal to one. We repeat this process for all N of the modified frequency regions.

Figure 8a shows that removing any of the N modified frequency regions results in a decrease to the overall fitness of the parameter set. The small change in fitness seen after the ablation of either region five or region six is consistent with the diminishing fitness returns seen in Figure 7, and shows how each modified frequency region contributes to the overall privacy protections offered by SafeSpeaker. The results of the ablation study, along with the results of the evaluation of the number of modified frequency regions shown in Figure 7 indicate



(a) Percent change in fitness caused by ablating a given frequency region.

Region #.	Lower Bound	Upper Bound	Coeff. (a_k)
1	1886	6012	1.17
2	1423	7039	0.14
3	606	6089	1.38
4	2959	5123	1.83
5	1386	6366	1.23
6	2029	7133	1.48

(b) The obfuscation parameters used during the ablation study.

Fig. 8. The impact that ablating each of the N modified frequency regions has on the overall fitness of the parameter set. Ablating region k is equivalent to setting $a_k = 1$.

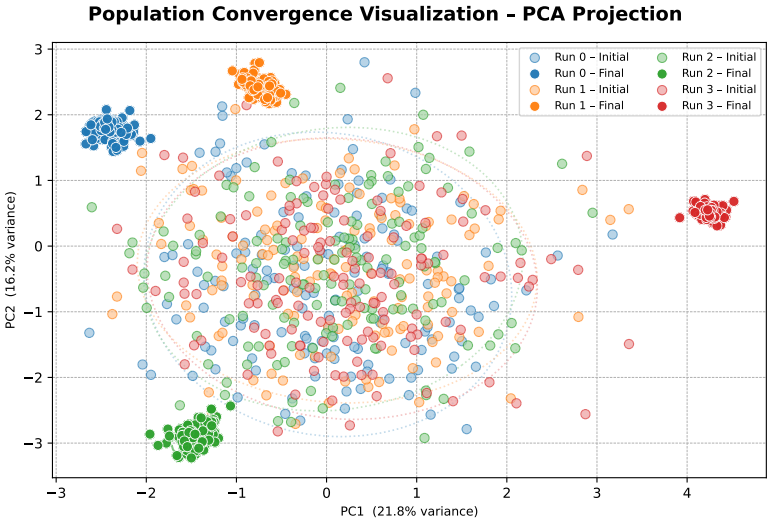
that it may be possible to achieve similar privacy protections with fewer bands, which would decrease the overall computational requirements of the obfuscation algorithm and allow privacy protections to extend to an even wider range of resource-constrained devices. Here our evaluation uses six modified frequency regions in order to achieve the greatest level of privacy protection that is available on our chosen hardware platform.

7.3.3 Seeded Optimization. To investigate how well the parameter optimization process can explore the search space and evolve towards high-fitness solutions, we performed a number of optimization passes with different starting seeds. We chose to use four different starting seeds in our evaluation given the compute time required to perform the full optimization. The optimization configuration was identical to the setup discussed in Section 7 with the exception of the starting seed. The sum of the total used wall-clock time across all four optimizations was roughly 26.6 days.

Figure 9a shows the initial and final populations of the four optimization passes as a PCA projection. Visualizing the high-dimensional individuals as a PCA projection highlights patterns that emerge both across and within the four optimizations. The lighter colored dots correspond to the initial populations while the darker colored dots represent members of the final generation. The PCA projection confirms that the initial population is spread out across the search space and accounts for a wide variety of potential solutions. Furthermore, the projection shows that the final population from each run is able to converge on a high-fitness solution. It is interesting to note that the different optimization passes produce distinct solutions. The average population fitness and best individual fitness are shown in Table 9b. Despite the differences in the optimized solutions, all four optimized parameter sets achieve similar fitness. As shown later in Section 8.2.3, using multiple parameter sets mitigates the risks posed by more capable attackers. Although we limit our evaluation to four sets of parameters, future work may explore the effects of higher numbers of obfuscation parameter sets.

8 Obfuscation Evaluation

In this section, we evaluate the privacy-utility trade-off of audio obfuscated by SafeSpeaker. To compare with related work, we implemented three existing DSP-based voice obfuscation techniques: VoiceMask [41], the



(a) PCA projection of the initial and final populations of four seeded optimization passes.

Run #.	Avg. Fitness	Max Fitness
1	1.74	1.76
2	1.82	1.85
3	1.78	1.81
4	1.79	1.82

(b) Fitness statistics for the optimization passes.

Fig. 9. PCA projection and fitness statistics for the multiple seeded optimization passes. The PCA projection shows that the random initial population explores the search space and converges on high-fitness solutions.

McAdams transformation [40], and MicPro [60]. The evaluation is conducted on four metrics: equal error rate (EER), true positive rate (TPR), word error rate (WER), and obfuscation latency.

8.1 Evaluation Setup

Provided below are additional details about the metrics, datasets, models, and other obfuscation systems that were used as a part of the evaluation process. Table 1 shows the set of obfuscation parameters that are evaluated in this section.

8.1.1 Datasets. To evaluate the performance of SafeSpeaker across a variety of speaker conditions, we leverage two commonly used speech datasets: the LibriSpeech corpus [39] and the voice cloning toolkit corpus (VCTK) [61]. The EER, TPR, WER, and obfuscation latency evaluation are conducted using the LibriSpeech-Test [39] and VCTK-Test [61] subsets. The parameter optimization discussed in Section 7 is performed using the VCTK-Dev subset. The custom ASV models used in the adaptive attacker evaluation discussed in Section 8.2.3 are trained on

Region #.	Lower Bound	Upper Bound	Coeff. (a_k)
1	711	7999	1.29194
2	5490	6436	0.96743
3	1849	3079	0.14014
4	572	7999	1.44165
5	935	7033	1.17383
6	3040	7098	0.91

Table 1. The obfuscation parameters used during the evaluation of SafeSpeaker.

the LibriSpeech train-clean-360 subset. Detailed information on the construction of each dataset is displayed in Table 2.

Table 2. Datasets used for evaluation

Dataset	Male Speakers	Female Speakers	Length (Hours)	Utterance Count
VCTK dev	15	15	11.00	12253
VCTK test	15	15	11.38	12350
LibriSpeech test-clean	20	20	5.40	2620
LibriSpeech train-clean-360	482	439	363.6	104014

8.1.2 Evaluation Models. The privacy and usability evaluations presented in this section were performed using five different pre-trained models. Three different ASV models were used to evaluate the EER and TPR: an ECAPA-TDNN model trained on LibriSpeech train-clean-360 [51], an XVector-based model trained on the VoxCeleb dataset [43, 46], and a ResNet-TDNN model trained on the VoxCeleb dataset [43, 58]. Two different ASR models were used for the WER evaluation: a wav2vec2-based model provided by the Voice Privacy Challenge 2024 [51] and OpenAI's whisper-base [42]. Detailed information on each of these models is provided in Table 3.

In addition, the adaptive attacker ASV models used in Section 8.2.3 were constructed by retraining the ECAPA-TDNN ASV model on obfuscated versions of the LibriSpeech train-clean-360 dataset.

Table 3. ASV and ASR models used for evaluation

ASV Architecture	Training Data	Source	ASR Architecture	Training Data	Source
ECAPA-TDNN	LibriSpeech train-clean-360	VPC2024 [51]	Wav2Vec2	LibriSpeech train-960	VPC2024 [51]
X-Vector / TDNN	VoxCeleb1+2	SpeechBrain [43]	Whisper-Base	680k hours of internet audio	OpenAI [42]
ResNet-TDNN	VoxCeleb1+2	SpeechBrain [43]			

8.1.3 Baseline Systems. The existing DSP-based obfuscation techniques are implemented according to the details provided by their respective authors. For techniques with variable obfuscation parameters, we chose the parameters presented by the authors that best fit the evaluation.

VoiceMask [41], initially discussed in Section 4.2.1, uses an obfuscation technique that is based on vocal tract length normalization (VTLN). In their work, the authors evaluated multiple frequency warping functions. The implementation used in this evaluation selects the frequency warping function that the authors indicated achieved the best performance.

The McAdams transformation [40], initially discussed in Section 4.2.2, uses a formant modification approach based on linear predictive coding. The obfuscation parameter, α, can be randomized within a pre-determined range. In the evaluation presented in this section, we choose $\alpha \in U(0.5, 0.9)$, the range that achieved the highest level of privacy protection in the evaluation presented in [40].

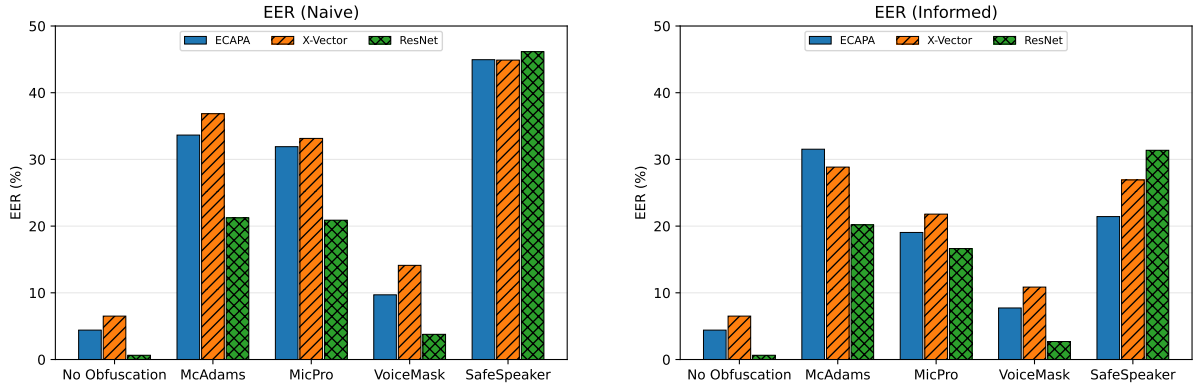
MicPro [60], initially discussed in Section 4.2.3, uses a parameter search to find a set of obfuscation parameters. This parameter search was performed using the process provided by the authors in their publicly available code repository. MicPro uses a process that requires audio to be sampled at 8 kHz, but the audio used in the evaluation presented in this section is sampled at 16 kHz. To account for this, utterances obfuscated with MicPro were downsampled to 8 kHz to maintain compatibility with the original implementation. When evaluating obfuscation latency, operating on audio sampled at 8 kHz rather than 16 kHz results in a shorter obfuscation latency, as there

are fewer samples to process. To account for this, the measured obfuscation latency of MicPro was multiplied by two to account for the difference in the number of samples.

8.2 Privacy Evaluation

The equal error rate (EER) and the true positive rate (TPR) are used to evaluate the privacy of obfuscated audio.

8.2.1 Equal Error Rate. The EER evaluation presented in this section is conducted using three pre-trained ASV models. These models include an ECAPA-TDNN model provided by the 2024 Voice Privacy Challenge [51], SpeechBrain's spkrec-xvect-voxceleb [43], and SpeechBrain's spkrec-resnet-voxceleb [43]. More information on these models can be found in Table 3. Similarly to the EER evaluation performed during parameter optimization in Section 7.1, both the naïve EER (EER_{naive}) and the informed EER (EER_{inf}) were used to evaluate the privacy of obfuscated audio.



(a) Average naïve EER.

(b) Average informed EER.

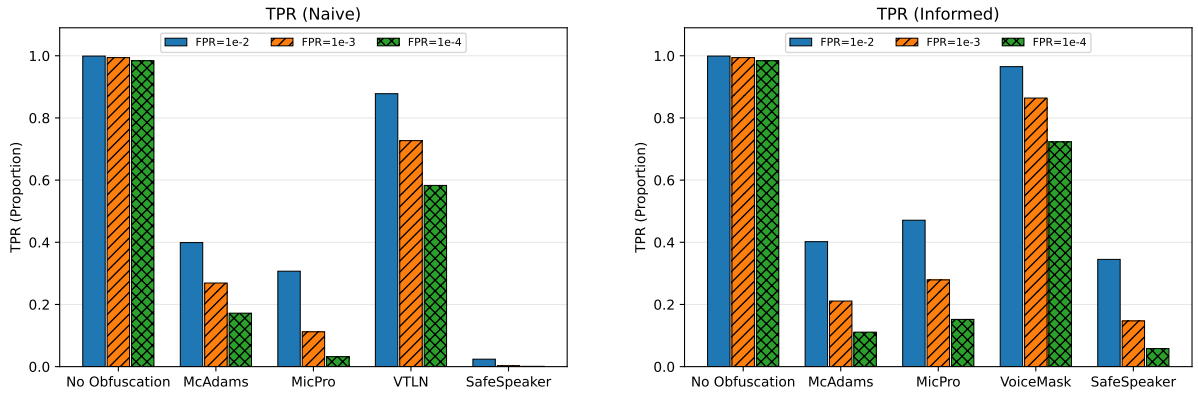
Fig. 10. Overall results of the EER portion of the privacy evaluation.

Figure 10a displays the average EER_{naive} achieved by each obfuscation system across all datasets. In the naïve case, SafeSpeaker provides a stronger level of privacy protection across all three ASV models compared to the existing resource-aware obfuscation methods. Figure 10b displays the average EER_{inf} achieved by each obfuscation system across all datasets. Although all of the evaluated obfuscation systems experience a drop in performance when the attacker is aware of the obfuscation method and parameters, SafeSpeaker still provides the strongest privacy protection for two of the three ASV models. A detailed breakdown of the EER_{naive} and EER_{inf} results is provided in Table A1a and Table A1b, respectively.

8.2.2 True Positive Rate. Recent work has shown that evaluating the privacy of voice obfuscation solely according to the EER risks overestimating the strength of privacy protection [38, 55]. In addition to evaluating the privacy of obfuscated audio according to the EER, we also evaluated the true positive rate (TPR) of each ASV model at low false positive rate (FPR) values (FPR $\in \{.01, .001, .0001\}$). Whereas the EER represents the performance of a balanced ASV system, the TPR at low FPR values represents the performance of an ASV system meant to robustly reject impostors. Lower TPR values represent stronger privacy protection.

The TPR at a given FPR value is determined by modifying the similarity threshold, θ, that the ASV model uses to decide if two speakers are the same. The TPR is calculated by finding the value of θ at which the proportion of non-target speaker utterances that are falsely accepted is equal to the desired FPR value. The corresponding TPR

value is then measured by computing the proportion of target speaker utterances that are correctly accepted at that value of θ. In case no value of θ exactly produces the target FPR, the value of the TPR is calculated by interpolation. Similarly to the EER, the TPR is calculated for both the naïve and informed attacker scenarios. The evaluation in this section reports only the highest TPR value measured across all three ASV models for each target FPR; this TPR value represents the worst-case privacy protection.



(a) Average naïve TPR.

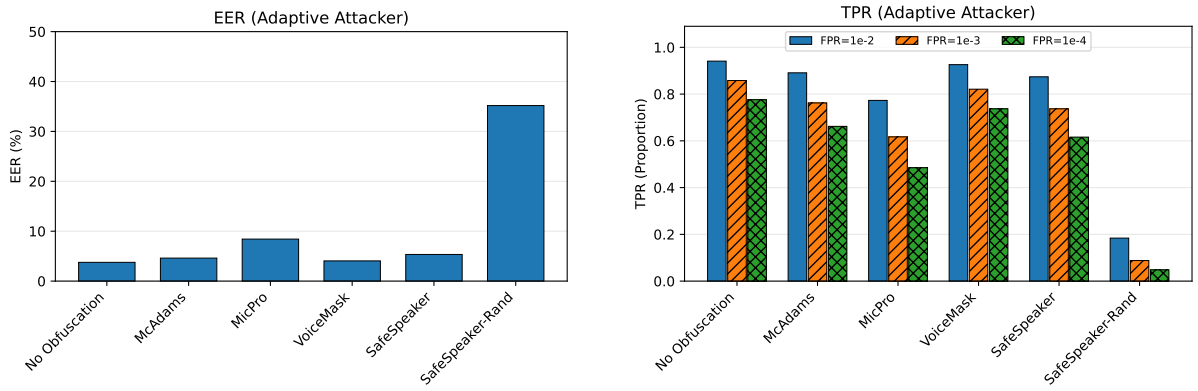
(b) Average informed TPR.

Fig. 11. Overall results of the TPR portion of the privacy evaluation.

Figure 11a displays the highest average TPR across the three ASV models that is achieved by each obfuscation system in the naïve attacker scenario. In the naïve case, SafeSpeaker achieves the lowest TPR at all FPR values, outperforming all of the other resource-aware obfuscation methods. Figure 11b displays the average maximum TPR achieved by each obfuscation system in the informed attacker scenario. Similarly to the EER evaluation, all of the evaluated obfuscation systems experience a drop in performance when the attacker is aware of the obfuscation method and parameters. However, SafeSpeaker still achieves the lowest TPR amongst all of the obfuscation methods. A detailed breakdown of the naïve TPR and the informed TPR results are provided in Table A2a and Table A2b, respectively.

8.2.3 Adaptive Attacker Scenario. The evaluations presented in Section 8.2.1 and Section 8.2.2 consider attackers that do not attempt to adapt their attacks to the obfuscation system. To provide a full picture of the privacy protections offered by a voice obfuscation system, it is important to consider an attacker that attempts to use their knowledge of the obfuscation process to modify their attack. Existing work regards attackers that retrain ASV models on obfuscated audio to be the strongest form of attacker [51, 53]. To evaluate this type of adaptive attacker, we trained four ECAPA-TDNN based ASV models, one for each of the obfuscation methods evaluated in this section. LibriSpeech train-clean-360 was obfuscated using each method to create the training data for each of the four models.

We also introduce a new form of SafeSpeaker, referred to as SafeSpeaker-Rand, that is designed to mitigate the privacy threats posed by adaptive attackers. SafeSpeaker-Rand changes the obfuscation parameters between utterances in order to reduce the risks associated with using a static set of obfuscation parameters. The additional obfuscation parameter sets were constructed by performing the optimization process discussed in Section 7 with different starting seeds. Three additional sets of optimized obfuscation parameters were produced for the adaptive attacker evaluation.



(a) Average adaptive EER.

(b) Average adaptive TPR.

Fig. 12. EER and TPR results of the adaptive attacker evaluation.

Figures 12a and 12b show the EER and TPR when the attacker has trained the ASV model on obfuscated speech. Compared to the informed attacker results shown in Figures 10b and 11b, the adaptive attacker is better able to defeat the obfuscation. SafeSpeaker-Rand shows how the introduction of additional obfuscation parameter sets mitigates the risk posed by adaptive attackers. SafeSpeaker and SafeSpeaker-Rand provide the strongest level of privacy protection among the resource-aware methods.

8.3 Usability Evaluation

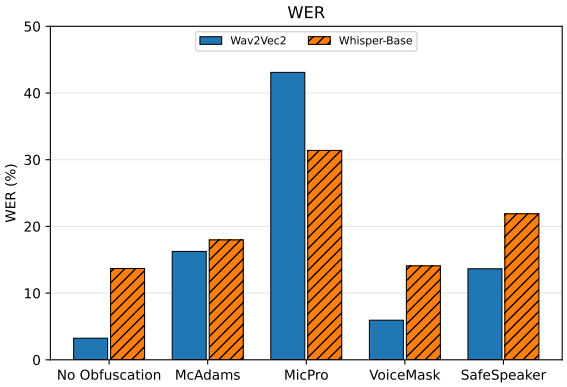
The usability of SafeSpeaker was evaluated using the word error rate and the obfuscation time.

8.3.1 Word Error Rate. The WER evaluation in this section is done with two pre-trained ASR models. These models include the wav2vec2 model provided by the 2024 Voice Privacy Challenge [51] and OpenAI's whisper-base [42]. More information on these models can be found in Table 3. Section 7.1.2 details how the WER was determined.

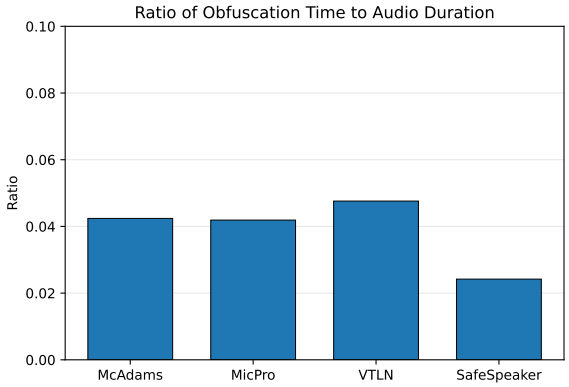
Figure 13a compares the average WER achieved by each obfuscation method across the evaluation datasets. Industry standards indicate that an average WER of less than 20% is acceptable for a voice interface [57], which SafeSpeaker is able to achieve. This indicates that audio obfuscated using SafeSpeaker will maintain an acceptable level of usability with existing transcription models.

8.3.2 Computational Overhead. To measure the computational overhead of the obfuscation process, the anonymization framework used for the privacy and usability evaluations was modified to measure the elapsed time. To provide a fair estimate of the obfuscation time, the evaluation uses a Python-based implementation of each technique to match the language chosen for the evaluation toolkit. To reduce the impact of Python overhead on the obfuscation time, the evaluation uses just-in-time compilation to compile the Python code at runtime. All obfuscation techniques were restricted to a single CPU core to avoid timing discrepancies. The evaluation is performed using a 12th generation Intel i7 processor and 128GB of RAM. The utterances used for the timing evaluation were taken from the LibriSpeech dataset [39] and total 82 seconds.

Figure 13b shows the average time for SafeSpeaker and the three other obfuscation systems to modify the fixed set of utterances as a ratio of obfuscation time to audio time. SafeSpeaker achieves a faster obfuscation time



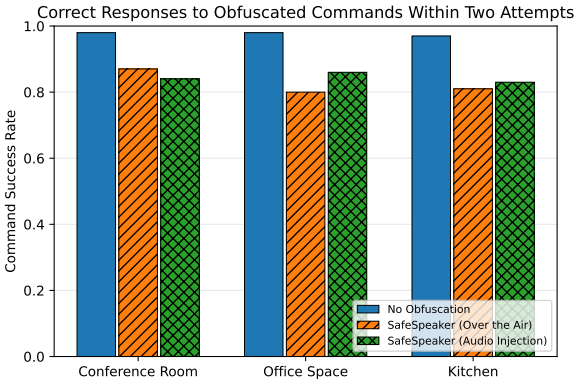
(a) Average WER.



(b) Ratio of obfuscation time to utterance duration for the proposed method and each of the existing DSP-based systems taken from existing work.

Fig. 13. Overall results of the WER and Obfuscation latency portions of the usability evaluation.

compared to the other resource-aware methods, enabling obfuscation-based privacy to be expanded to a larger class of resource-constrained voice-interfaces and IoT devices.



(a)

Environment	Ambient Sound Level (dB)	RT60 (s)
Conference Room	42.8	0.58
Office Space	51.0	0.72
Kitchen	48.3	0.32

(b)

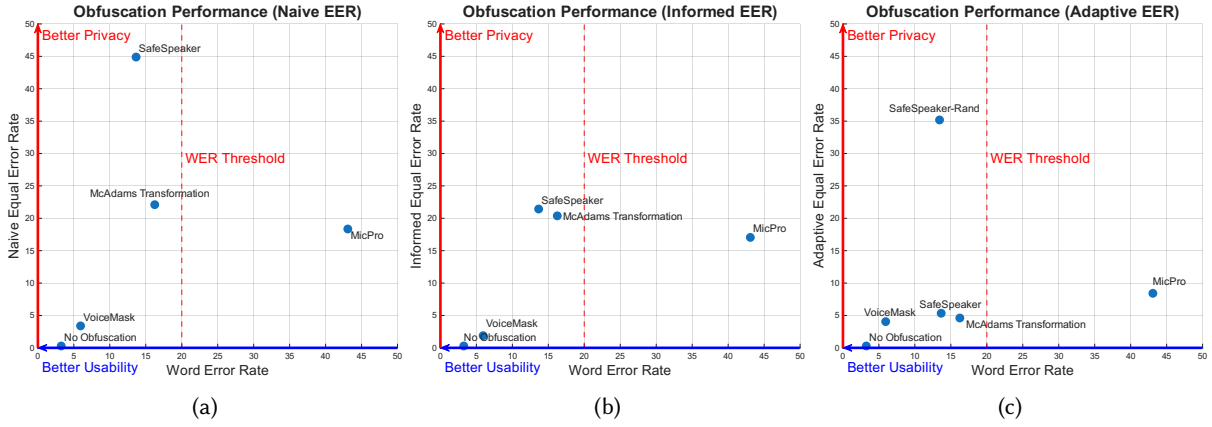
Fig. 14. 14a: the proportion of the 100 obfuscated commands that received a correct response in each environment. The proportion of commands that received a correct response on the first attempt is shown in blue and the proportion of commands that received a correct response within two attempts is shown in hatched orange. The table in Figure 14b shows the details of the testing environments.

8.3.3 Real-World Evaluation. The WER evaluation presented in Section 8.3.1 provides insight into how SafeSpeaker impacts the intelligibility of modified audio. However, perfect intelligibility is not always required for an efficient voice interface. The obfuscation algorithm presented in this work is designed to be compatible with a large variety of voice obfuscation systems on resource-constrained IoT devices, such as smart speakers or

vehicle voice control systems. These systems are often able to produce correct responses even when the spoken command isn't perfectly understood.

To better understand how audio obfuscated with SafeSpeaker maintains usability with voice-enabled IoT devices, we evaluated how an off-the-shelf smart speaker responded to obfuscated audio commands. A total of 100 smart speaker commands from five male speakers and five female speakers were taken from the fluent-speech-corpus [26], obfuscated, and played to an off-the-shelf smart speaker. The obfuscated commands were played using two methods. In the first, obfuscated audio was played over the air using a Bluetooth speaker. In the second, a microcontroller captures clean audio emitted from a Bluetooth speaker, obfuscates the audio in real-time, and injects the obfuscated audio into the digital audio bus of the smart speaker. To evaluate how different acoustic conditions impact the obfuscated audio, this test was repeated in a total of three distinct environments: A quiet conference room, a typical office environment, and a typical kitchen environment. Table 14b shows the measured acoustic conditions of the three different test environments; the measurements for the three locations were taken using a smartphone. Figure 14a compares the proportion of commands that received correct responses within two attempts. On average 83% of the commands obfuscated by SafeSpeaker received a correct response within two attempts when played over the air and 84% of the commands received a correct response within two attempts when the obfuscated audio is injected into the smart speaker. From our observation of the results, speaker accent and fast speaking rate were common attributes among failed commands. These results show that SafeSpeaker is able to maintain the usability of existing voice interfaces while providing users with robust privacy protections.

8.4 Overall Performance



(a) (b) (c)

Fig. 15. Scatter plots of the EER and WER achieved by various DSP-based voice obfuscation systems. Figure 15a shows the naïve evaluation scenario, Figure 15b shows the informed evaluation scenario, and Figure 15c shows the adaptive scenario. Systems with better privacy protections are found towards the top of the plot and systems with better usability are found towards the left side of the plot.

The proposed method is able to significantly increase the EER of a speaker verification system when compared to no obfuscation. Figure 15 shows the EER and WER achieved by SafeSpeaker compared to the three existing DSP-based obfuscation techniques. The left plot shows the tradeoff between privacy and usability when considering the naïve EER and the right plot shows the same tradeoff for the informed EER. The best obfuscation techniques occupy the upper-left portion of the plots, which corresponds to a high level of privacy protection with minimal impact on audio usability. The dashed line represents the WER threshold defined by [57]. The increased EER seen

in both the naïve and informed cases shows that the proposed obfuscation strategy can reduce the likelihood of a user's identity being inferred from their voice. SafeSpeaker provides the highest level of privacy protection in the naïve scenario while still achieving a minimal impact on the intelligibility of the obfuscated audio. In the case of the informed EER, SafeSpeaker and the McAdams transformation both achieve similarly high levels of privacy protection while still producing usable audio. Overall, SafeSpeaker achieves comparable performance in both EER tasks while producing audio that is still compatible with downstream voice services.

9 Hardware Implementation

The goal of the proposed obfuscation method is to create a voice obfuscation technique that is compatible with the resource-constrained hardware used for IoT voice interfaces. This section details the design of a microcontroller (MCU) implementation of the obfuscation algorithm proposed in this work. The goal of the hardware artifact is to show that the proposed algorithm is able to run in real-time on constrained hardware and obfuscate recorded audio. This section depicts the hardware artifact as additional hardware used in conjunction with an off-the-shelf smart speaker, but the proposed obfuscation algorithm could also be included as trusted software on an IoT microphone, be included as a privacy co-processor, or be integrated as a part of the digital microphones used in consumer devices.

9.1 Microcontroller Implementation

The STM32G431KB was chosen for the hardware implementation of SafeSpeaker. This microcontroller-class device demonstrates how voice obfuscation can be added to resource-constrained hardware for as little as 3 USD. The MCU development kit was used to facilitate developing the prototypes and to allow easy access to the MCU's peripherals and programming interface. The development kit is housed in a custom printed circuit board (PCB) designed to improve the interface between the MCU and the smart speaker used for the prototype. Figure 16 shows an image of the completed system.

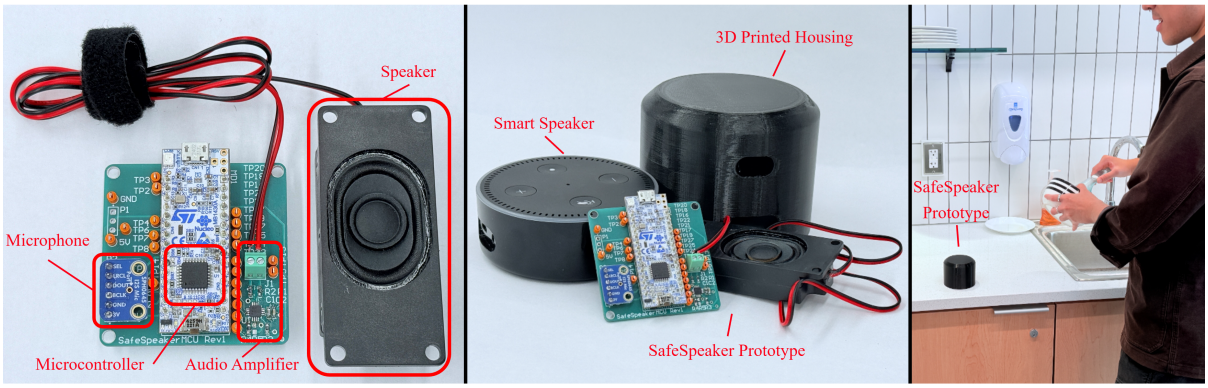


Fig. 16. Left: The printed circuit board (PCB) that was designed for the hardware prototype of SafeSpeaker. The individual components are labeled for clarity. Middle: The off-the-shelf smart speaker, printed circuit board, and 3D printed housing shown disassembled. Right: A demonstration of the form factor of SafeSpeaker in a kitchen environment.

Audio is input to the voice obfuscation algorithm through one of the MCU's Inter-IC Sound (I2S) peripheral interfaces and the MCU's direct memory access (DMA) controller. Using the DMA controller to handle input audio data offloads the task from the main processing element, allowing data collection and processing to be parallelized. The input audio is one channel (mono audio) sampled at 16 kHz with a bit depth of 16. After being

collected, the audio is converted from unsigned 16-bit integer to Q31 fixed-point format. Using a fixed-point type improves results in more efficient processing for the voice obfuscation algorithm on the STM32G431KB than can be achieved through a floating-point type. Once in the proper format, the audio is filtered using the fixed-point, cascaded biquadratic filtering functions of the ARM-CMSIS DSP library. The filter coefficients are pre-computed from the results of the parameter optimization described in Section 7.

After filtering, the resulting filtered audio bands are converted back to a signed 16-bit integer representation. This conversion greatly decreases the memory requirements of the obfuscation pipeline, as a persistent buffer is maintained for each of the audio bands processed by the algorithm. The frequency modifications for each band are then performed as described in Section 6. After modification, the resulting audio segments are combined and output to a second I2S interface through the DMA controller. The output I2S interface is connected to an audio amplifier and speaker, allowing the obfuscated audio to be played over the air. The MCU implements the obfuscation pipeline using a double-buffer technique so that new audio can be collected while the previous chunk is being processed. In addition to outputting audio over the air through an I2S speaker, we also developed a version of the hardware prototype that interfaces directly with the smart speaker. This version of the hardware prototype outputs the obfuscated audio using the STM32G431KB's serial audio interface (SAI) peripheral. The output audio is injected into the digital audio bus of the smart speaker, which had been modified to disconnect the original microphones. In this version, the hardware prototype acts as a privacy co-processor that obfuscates audio before it is recorded by the smart speaker. The audio injection version of the prototype is the one used in the evaluation found in Figure 14a.

9.2 Hardware Prototype Evaluation

To investigate how the privacy protections provided by SafeSpeaker can extend to real-world IoT devices, we performed a number of evaluations using the hardware prototype introduced in Section 9. The system specifications evaluation includes measurements of the obfuscation latency, memory usage, and the energy consumption of our hardware prototype while running the obfuscation algorithm.

9.2.1 System Characterization. To characterize the specification of the hardware prototype of SafeSpeaker, we measured the obfuscation latency, memory usage, and energy consumption. During each measurement, the hardware prototype was recording and obfuscating background sounds from the shared workspace in which the measurements were taken. Table 4 displays the results of each of the measurements discussed below.

Table 4. Summary of hardware prototype characterization

Obfuscation Latency (ms)	CPU Utilization (%)	Memory Usage (kB)	Current Consumption (mA)
9.724	30.45	30.48	34

9.2.2 Obfuscation Latency. To measure the obfuscation latency, the hardware prototype was configured to send a signal at the beginning and the end of the obfuscation process using a GPIO pin. The GPIO pin was pulled high at the start of obfuscation and pulled low once the obfuscation had finished. To show the CPU utilization, the hardware prototype was also configured to toggle a separate GPIO pin when a new chunk of audio had been recorded. The utilization is determined by measuring the percentage of the time between new audio chunks during which the obfuscation is running.

Figure 17 shows the oscilloscope measurements that were used to measure the obfuscation time and CPU utilization of the hardware prototype. As shown in the figure, the audio obfuscation begins immediately after

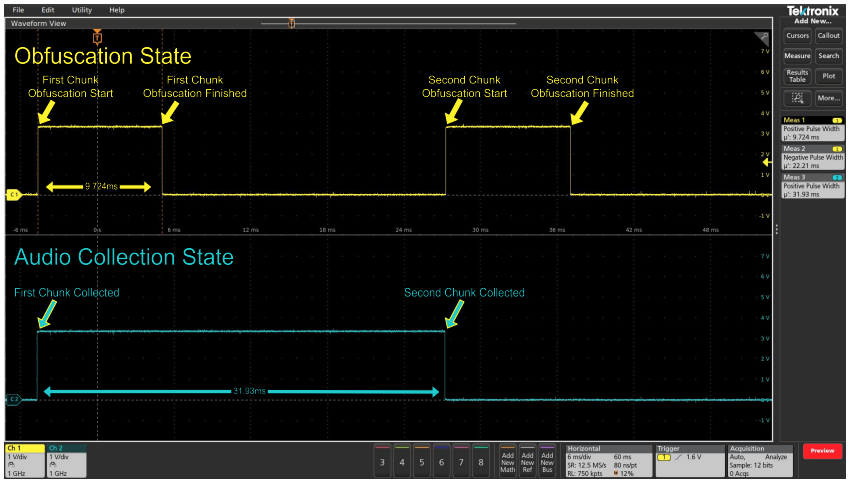


Fig. 17. Screen capture of the oscilloscope setup used to measure the obfuscation latency and CPU utilization of the obfuscation algorithm on the hardware prototype. The annotations detail the locations of the obfuscation signals, audio collection signals, and timings.

the first chunk of audio has been collected and finished 9.724 ms later. When the next chunk of audio has been collected, this process repeats. Overall, the audio obfuscation constitutes 35.45% of the processing time.

9.2.3 Memory Usage. The memory usage was calculated by compiling the firmware for the hardware prototype with additional compiler flags to output the static RAM and stack usage. The static RAM usage was measured from the compiled `.elf` file and the stack usage was measured from the compiled `.su` file. The total memory usage was 30480 bytes.

9.2.4 Energy Consumption. The energy consumption was measured using a source meter configured in ammeter mode. The STM32G431KB board used for the hardware prototype includes a jumper for measuring the current consumption of the microcontroller in isolation from the other hardware peripherals. By measuring the current across this jumper, we were able to evaluate the energy consumption of the obfuscation algorithm on the STM32G431KB. During the current measurement, the operating voltage was 3.3 V. The average current consumption over 796 measurements taken during the two-minute trial was 34 mA.

10 Discussion

This work presents a voice obfuscation scheme that is capable of both running in real time with the resources of a microcontroller-class device and providing privacy protections comparable to those of other DSP-based obfuscation schemes. The evaluations discussed above focus on smart speakers, as they are a common example of an IoT device with a voice interface, but this work aims to show how obfuscation can be used with a wider variety of resource-constrained devices. Voice interfaces are a convenient interaction modality for consumer smart devices because they take up less physical space compared to a full keyboard or touchscreen and can be manufactured at a fraction of the cost. However, simply recording and uploading raw audio creates privacy risks for the user. Voice obfuscation is an important privacy measure in such cases, mitigating privacy risks that are created by uploading recorded audio. Decreasing the computational overhead associated with voice obfuscation enables the use of obfuscation with a wider range of embedded voice interfaces. The resource-aware technique proposed in this work is designed to be compatible with low-cost, resource-constrained voice interfaces, such as

those that may be found in smart TV remotes or voice-enabled light switches, but can also be integrated at low cost into smartphones or laptops.

In addition to achieving a competitive level of privacy protection when compared to existing signal-processing-based obfuscation techniques, this work also provides those protections while maintaining the usability of the voice interface. The results from Figures 15 and 14a show that transcription accuracy is maintained within an acceptable margin (WER $\leq$ 20%) and that 97% of obfuscated commands are properly understood within two attempts. Although ML-based audio obfuscation is generally able to provide better privacy protections, the increases come at the cost of greatly increased computational overhead. Whereas machine learning may be appropriate for a server-class compute system that has access to ample processing power, such techniques are not compatible with real-time, resource-constrained IoT-scale systems. In contrast, the obfuscation technique presented in this work can be integrated into an embedded voice interface for less than $3 USD.

10.1 Limitations

As mentioned in Section 2, SafeSpeaker does not aim to protect against the leakage of broad speaker characteristics (e.g., gender, accent, or age) or private information that is explicitly revealed by the user. Although our evaluations show that SafeSpeaker robustly obfuscates an individual's identity, the obfuscation is not guaranteed to prevent attackers from learning such information.

Another limitation faced by this work is the computationally expensive fitness evaluation required during the optimization process. The optimization process relies on large speech datasets and high-resource ASV and ASR models to calculate the fitness of the candidate solutions. In our envisioned application, device manufacturers perform the optimization on a development corpus and package the resulting obfuscation parameter sets with the device firmware.

Another limitation of the work presented here arises from the evaluations taking place in a limited number of acoustic conditions. Although steps were taken to ensure the inclusion of multiple acoustic environments in our evaluation, the intelligibility and privacy metrics of the obfuscation method presented here may be susceptible to environmentally determined factors not considered in this work. As with traditional voice interfaces, additional engineering effort may be capable of reducing the impacts of environmental factors, but the aim of this work was to show that privacy protections can be achieved without major impacts to performance even on resource-constrained devices. Furthermore, the evaluations are only performed using English language utterances.

11 Conclusion

This work introduces a novel, signal-processing based architecture for performing voice obfuscation on resource-constrained devices. The resource-aware audio modification algorithm is intentionally designed to minimize computational overhead in an effort to enable real-time privacy protection on a wide range of voice interfaces, from embedded IoT devices to desktop-class computers. The algorithm is optimized through the use of an evolutionary algorithm with the goal of balancing the privacy and utility of voice interfaces. The results of our evaluations show that our method is capable of achieving both privacy protections and intelligibility ratings on par with those of similar signal-processing-based audio obfuscation techniques. Furthermore, we show that the technique presented here is more computationally efficient than approaches that achieve a similar level of protection, which enables audio obfuscation to be used at lower monetary cost and on a wider variety of devices. Overall, we show that signal-processing based audio obfuscation can be used to protect a user's identity while they interact with a voice interface for as little as $3 USD.

Acknowledgments

We would like to thank Shreyas Narayanan and Frank Anderson for their assistance related to hardware prototype modifications and data collection.

References

[1] Noura Abdi, Kopo M. Ramokapane, and Jose M. Such. 2019. More than Smart Speakers: Security and Privacy Perceptions of Smart Home Personal Assistants. In *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*. USENIX Association, Santa Clara, CA, 451–466. https://www.usenix.org/conference/soups2019/presentation/abdi

[2] Masato Akagi and Taw Ienaga. 1995. Speaker individualities in fundamental frequency contours and its control. In *4th European Conference on Speech Communication and Technology (Eurospeech 1995)*. 439–442. doi:10.21437/Eurospeech.1995-119

[3] Amazon. 2024. *Create an Alexa Voice ID.* Retrieved May 28, 2024 from https://www.amazon.com/gp/help/customer/display.html?nodeId=GYY4637XC2STFL9R

[4] Amazon. 2024. *Learn How Alexa Works - How Does Alexa Work?* Retrieved May 28, 2024 from https://www.amazon.com/b/?node=23608618011&tag=googhydr-20&hvadid=661712030105&hvpos=&hvnetw=g&hvrand=6070453287798683794&hvpone=&hvptwo=&hvqmt=e&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9016851&hvtargid=kwd-2141785072047&ref=pd_sl_4gozlak3z2_e&gclid=CjwKCAjwgdayBhBQEiwAXhMxtn0gu4_6XbsSpe-2twSPT1O9CYkykcpAsQdRj-rs2mWo95WU0rh7ABoCI_UQAvD_BwE

[5] Apple. 2024. *Set up voice recognition on HomePod or HomePod mini.* Retrieved May 28, 2024 from https://support.apple.com/en-us/108397#:~:text=On%20your%20iPhone%20or%20iPad%2C%20go%20to%20Settings%20%3E%20Siri%20%26,same%20language%20as%20your%20HomePod.

[6] Pierre Champion. 2023. Anonymizing speech: Evaluating and designing speaker anonymization techniques. *arXiv preprint arXiv:2308.04455* (2023).

[7] Varun Chandrasekaran, Suman Banerjee, Bilge Mutlu, and Kassem Fawaz. 2021. PowerCut and Obfuscator: An Exploration of the Design Space for Privacy-Preserving Interventions for Smart Speakers. In *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*. USENIX Association, 535–552. https://www.usenix.org/conference/soups2021/presentation/chandrasekaran

[8] Meng Chen, Li Lu, Junhao Wang, Jiadi Yu, Yingying Chen, Zhibo Wang, Zhongjie Ba, Feng Lin, and Kui Ren. 2023. VoiceCloak: Adversarial Example Enabled Voice De-Identification with Balanced Privacy and Utility. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 2, Article 48 (June 2023), 21 pages. doi:10.1145/3596266

[9] J. S. Chung, A. Nagrani, and A. Zisserman. 2018. VoxCeleb2: Deep Speaker Recognition. In *INTERSPEECH*.

[10] Raphael Davidian. 2017. *Alexa and Third Parties' Reasonable Expectation of Privacy.* American Criminal Law Review Online. https://www.law.georgetown.edu/american-criminal-law-review/aclr-online/volume-54/alexa-and-third-parties-reasonable-expectation-of-privacy/

[11] Matt Day, Giles Turner, and Natalia Drozdiak. 2019. *Thousands of Amazon Workers Listen to Alexa Users' Conversations.* Retrieved May 27, 2024 from https://time.com/5568815/amazon-workers-listen-to-alexa/

[12] F. Dellaert, T. Polzin, and A. Waibel. 1996. Recognizing emotion in speech. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP '96*, Vol. 3. 1970–1973 vol.3. doi:10.1109/ICSLP.1996.608022

[13] ecobee. 2025. *Smart Thermostat Premium.* Retrieved Aug 9, 2025 from https://www.ecobee.com/en-us/smart-thermostats/smart-thermostat-premium/

[14] Fuming Fang, Xin Wang, Junichi Yamagishi, Isao Echizen, Massimiliano Todisco, Nicholas Evans, and Jean-Francois Bonastre. 2019. Speaker Anonymization Using X-vector and Neural Waveform Models. arXiv:1905.13561 [eess.AS] https://arxiv.org/abs/1905.13561

[15] FBI San Francisco. 2024. *FBI Warns of Increasing Threat of Cyber Criminals Utilizing Artificial Intelligence.* Federal Bureau of Investigation. https://www.fbi.gov/contact-us/field-offices/sanfrancisco/news/fbi-warns-of-increasing-threat-of-cyber-criminals-utilizing-artificial-intelligence

[16] Asif A. Ghazanfar and Drew Rendall. 2008. Evolution of human vocal production. *Current Biology* 18, 11 (2008), R457–R460. doi:10.1016/j.cub.2008.03.030

[17] Google. 2024. *How Google Voice Works.* Retrieved May 27, 2024 from https://policies.google.com/technologies/voice?hl=en-US

[18] Google. 2024. *Turn on voice recognition with Voice Match.* Retrieved May 28, 2024 from https://support.google.com/assistant/answer/9071681?hl=en&co=GENIE.Platform%3DAndroid

[19] Scharon Harding. 2025. *Everything you say to your Echo will be sent to Amazon starting on March 28.* ArsTechnica. https://arstechnica.com/gadgets/2025/03/everything-you-say-to-your-echo-will-be-sent-to-amazon-starting-on-march-28/

[20] Alex Hern. 2019. *Apple contractors 'regularly hear confidential details' on Siri recordings.* Retrieved May 5, 2024 from https://www.theguardian.com/technology/2019/jul/26/apple-contractors-regularly-hear-confidential-details-on-siri-recordings

[21] Sean Hollister. 2022. *Today I learned Amazon has a form so police can get my data without permission or a warrant.* Retrieved Aug 31, 2025 from https://www.theverge.com/2022/4/28/23047026/amazon-alexa-voice-data-targeted-ads-research-report

[22] Gary Horcher. 2018. *Woman says her Amazon device recorded private conversation, sent it out to random contact.* Retrieved May 5, 2024 from https://www.kiro7.com/news/local/woman-says-her-amazon-device-recorded-private-conversation-sent-it-out-to-random-contact/755507974/

[23] Peter Ladefoged and D. E. Broadbent. 1957. Information Conveyed by Vowels. *The Journal of the Acoustical Society of America* 29, 1 (01 1957), 98–104. arXiv:https://pubs.aip.org/asa/jasa/article-pdf/29/1/98/18736057/98$_1$o$nline.pdf$ doi:10.1121/1.1908694

[24] Josephine Lau, Benjamin Zimmerman, and Florian Schaub. 2018. Alexa, Are You Listening? Privacy Perceptions, Concerns and Privacy-seeking Behaviors with Smart Speakers. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW, Article 102 (nov 2018), 31 pages. doi:10.1145/3274371

[25] Duc Le and Emily Mower Provost. 2013. Emotion recognition from spontaneous speech using Hidden Markov models with deep belief networks. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. 216–221. doi:10.1109/ASRU.2013.6707732

[26] Loren Lugosch, Mirco Ravanelli, Patrick Ignoto, Vikrant Singh Tomar, and Yoshua Bengio. 2019. Speech Model Pre-Training for End-to-End Spoken Language Understanding. In *Interspeech 2019*. 814–818. doi:10.21437/Interspeech.2019-2396

[27] Nathan Malkin, Joe Deatrick, Allen Tong, Primal Wijesekera, Serge Egelman, and David Wagner. 2019. Privacy Attitudes of Smart Speaker Users. *Proceedings on Privacy Enhancing Technologies* 2019 (10 2019), 250–271. doi:10.2478/popets-2019-0068

[28] Candy Olivia Mawalim, Shogo Okada, and Masashi Unoki. 2022. Speaker anonymization by pitch shifting based on time-scale modification. In *Proc. 2nd Symposium on Security and Privacy in Speech Communication*. 35–42. doi:10.21437/SPSC.2022-7

[29] Sarina Meyer, Florian Lux, Pavel Denisov, Julia Koch, Pascal Tilli, and Ngoc Thang Vu. 2022. Speaker anonymization with phonetic intermediate representations. *arXiv preprint arXiv:2207.04834* (2022).

[30] Sarina Meyer, Pascal Tilli, Pavel Denisov, Florian Lux, Julia Koch, and Ngoc Thang Vu. 2023. Anonymizing Speech with Generative Adversarial Networks to Preserve Speaker Privacy. In *2022 IEEE Spoken Language Technology Workshop (SLT)*. 912–919. doi:10.1109/SLT54892.2023.10022601

[31] Abraham Mhaidli, Manikandan Venkatesh, Yixin Zou, and Florian Schaub. 2020. Listen Only When Spoken To: Interpersonal Communication Cues as Smart Speaker Privacy Controls. *Proceedings on Privacy Enhancing Technologies* 2020 (04 2020), 251–270. doi:10.2478/popets-2020-0026

[32] Microsoft. 2024. *How does Microsoft protect my privacy while improving its speech recognition technology?* Retrieved May 27, 2024 from https://support.microsoft.com/en-us/windows/how-does-microsoft-protect-my-privacy-while-improving-its-speech-recognition-technology-f465d7a7-4a4f-40b7-9441-f0e6e97e24ec

[33] Arsha Nagrani, Joon Son Chung, Weidi Xie, and Andrew Zisserman. 2019. Voxceleb: Large-scale speaker verification in the wild. *Computer Science and Language* (2019).

[34] A. Nagrani, J. S. Chung, and A. Zisserman. 2017. VoxCeleb: a large-scale speaker identification dataset. In *INTERSPEECH*.

[35] NPR and EdisonResearch. 2022. *The Smart Audio Report.* Retrieved May 29, 2024 from https://www.nationalpublicmedia.com/insights/reports/smart-audio-report/

[36] State of California Department of Justice. 2024. *California Consumer Privacy Act (CCPA).* Retrieved Sep 10, 2024 from https://oag.ca.gov/privacy/ccpa

[37] National Institute of Standards and Technology. 2021. *Cybersecurity Labeling for Consumers: Internet of Things (IoT) Devices and Software.* Retrieved Sep 9, 2024 from https://www.nist.gov/itl/executive-order-14028-improving-nations-cybersecurity/cybersecurity-labeling-consumers-0

[38] Michele Panariello, Sarina Meyer, Pierre Champion, Xiaoxiao Miao, Massimiliano Todisco, Ngoc Thang Vu, and Nicholas Evans. 2025. The Risks and Detection of Overestimated Privacy Protection in Voice Anonymisation. In *5th Symposium on Security and Privacy in Speech Communication*. 8–12. doi:10.21437/SPSC.2025-2

[39] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 5206–5210. doi:10.1109/ICASSP.2015.7178964

[40] Jose Patino, Natalia Tomashenko, Massimiliano Todisco, Andreas Nautsch, and Nicholas Evans. 2021. Speaker Anonymisation Using the McAdams Coefficient. In *Proc. Interspeech 2021*. 1099–1103. doi:10.21437/Interspeech.2021-1070

[41] Jianwei Qian, Haohua Du, Jiahui Hou, Linlin Chen, Taeho Jung, and Xiang-Yang Li. 2018. Hidebehind: Enjoy Voice Input with Voiceprint Unclonability and Anonymity. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems* (Shenzhen, China) *(SenSys '18)*. Association for Computing Machinery, New York, NY, USA, 82–94. doi:10.1145/3274783.3274855

[42] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust Speech Recognition via Large-Scale Weak Supervision. doi:10.48550/ARXIV.2212.04356

[43] Mirco Ravanelli, Titouan Parcollet, Peter Plantinga, Aku Rouhe, Samuele Cornell, Loren Lugosch, Cem Subakan, Nauman Dawalatabad, Abdelwahab Heba, Jianyuan Zhong, Ju-Chieh Chou, Sung-Lin Yeh, Szu-Wei Fu, Chien-Feng Liao, Elena Rastorgueva, François Grondin, William Aris, Hwidong Na, Yan Gao, Renato De Mori, and Yoshua Bengio. 2021. SpeechBrain: A General-Purpose Speech Toolkit. arXiv:2106.04624 [eess.AS] arXiv:2106.04624.

[44] Emma Roth. 2022. *Apple and Meta shared data with hackers pretending to be law enforcement officials.* The Verge. https://www.theverge.com/2022/3/30/23003600/apple-meta-shared-data-hackers-pretending-law-enforcement-officials News article.

[45] Samsung. 2025. *22 cu. ft. Counter Depth Side-by Side Refrigerator with Touch Screen Family Hub in Stainless Steel.* Retrieved Aug 9, 2025 from https://www.samsung.com/us/home-appliances/refrigerators/side-by-side/22-cu-ft-counter-depth-side-by-side-refrigerator-with-touch-screen-family-hub-in-stainless-steel-rs22t5561sr-aa/

[46] David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. 2018. Spoken Language Recognition using X-vectors. In *The Speaker and Language Recognition Workshop (Odyssey 2018)*. 105–111. doi:10.21437/Odyssey.2018-15

[47] Brij Mohan Lal Srivastava, Natalia Tomashenko, Xin Wang, Emmanuel Vincent, Junichi Yamagishi, Mohamed Maouche, Aurélien Bellet, and Marc Tommasi. 2020. Design choices for x-vector based speaker anonymization. *arXiv preprint arXiv:2005.08601* (2020).

[48] Brij Mohan Lal Srivastava, Nathalie Vauquier, Md Sahidullah, Aurélien Bellet, Marc Tommasi, and Emmanuel Vincent. 2020. Evaluating voice conversion-based privacy protection against informed attackers. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2802–2806.

[49] Ke Sun, Chen Chen, and Xinyu Zhang. 2020. "Alexa, stop spying on me!": speech privacy protection against voice assistants. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems* (Virtual Event, Japan) *(SenSys '20)*. Association for Computing Machinery, New York, NY, USA, 298–311. doi:10.1145/3384419.3430727

[50] Seyed Mohammadjavad Seyed Talebi, Ardalan Amiri Sani, Stefan Saroiu, and Alec Wolman. 2021. MegaMind: a platform for security & privacy extensions for voice assistants. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services* (Virtual Event, Wisconsin) *(MobiSys '21)*. Association for Computing Machinery, New York, NY, USA, 109–121. doi:10.1145/3458864.3467962

[51] Natalia Tomashenko, Xiaoxiao Miao, Pierre Champion, Sarina Meyer, Xin Wang, Emmanuel Vincent, Michele Panariello, Nicholas Evans, Junichi Yamagishi, and Massimiliano Todisco. 2024. The VoicePrivacy 2024 Challenge Evaluation Plan. (2024). arXiv:2404.02677 [eess.AS]

[52] Natalia Tomashenko, Brij Mohan Lal Srivastava, Xin Wang, Emmanuel Vincent, Andreas Nautsch, Junichi Yamagishi, Nicholas Evans, Jose Patino, Jean-François Bonastre, Paul-Gauthier Noé, and Massimiliano Todisco. 2022. The VoicePrivacy 2020 Challenge Evaluation Plan. arXiv:2205.07123 [cs.CL]

[53] Natalia Tomashenko, Xin Wang, Xiaoxiao Miao, Hubert Nourtel, Pierre Champion, Massimiliano Todisco, Emmanuel Vincent, Nicholas Evans, Junichi Yamagishi, and Jean-François Bonastre. 2022. The VoicePrivacy 2022 Challenge Evaluation Plan. arXiv:2203.12468 [eess.AS]

[54] Natalia Tomashenko, Xin Wang, Xiaoxiao Miao, Hubert Nourtel, Pierre Champion, Massimiliano Todisco, Emmanuel Vincent, Nicholas Evans, Junichi Yamagishi, Jean-François Bonastre, and Michele Panariello. 2022. *The VoicePrivacy 2022 Challenge.* Retrieved May 27, 2024 from https://www.voiceprivacychallenge.org/results-2022/docs/VoicePrivacy_2022_Challenge___Natalia_Tomashenko.pdf

[55] Efthymios Tsaprazlis, Thanathai Lertpetchpun, Tiantian Feng, Sai Praneeth Karimireddy, and Shrikanth Narayanan. 2026. VoxGuard: Evaluating User and Attribute Privacy in Speech via Membership Inference Attacks. In *ICASSP 2026 - 2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 19042–19046. doi:10.1109/ICASSP55912.2026.11464033

[56] Jennifer Tuohy. 2022. *Researchers find Amazon uses Alexa voice data to target you with ads.* Retrieved May 27, 2024 from https://www.theverge.com/2022/4/28/23047026/amazon-alexa-voice-data-targeted-ads-research-report

[57] Eric Urban and Nitin Mehrotra. 2024. *Test accuracy of a custom speech model.* Retrieved May 27, 2024 from https://learn.microsoft.com/en-us/azure/ai-services/speech-service/how-to-custom-speech-evaluate-data?pivots=speech-studio

[58] Jesús Villalba, Nanxin Chen, David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Jonas Borgstrom, Leibny Paola García-Perera, Fred Richardson, Réda Dehak, Pedro A. Torres-Carrasquillo, and Najim Dehak. 2020. State-of-the-art speaker recognition with neural network embeddings in NIST SRE18 and Speakers in the Wild evaluations. *Computer Speech Language* 60 (2020), 101026. doi:10.1016/j.csl.2019.101026

[59] Ben Wolford. [n. d.]. *What is GDPR, the EU's new data protection law.* Retrieved Sep 10, 2024 from https://gdpr.eu/what-is-gdpr/

[60] Shilin Xiao, Xiaoyu Ji, Chen Yan, Zhicong Zheng, and Wenyuan Xu. 2023. MicPro: Microphone-based Voice Privacy Protection. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security* (Copenhagen, Denmark) *(CCS '23)*. Association for Computing Machinery, New York, NY, USA, 1302–1316. doi:10.1145/3576915.3616616

[61] Junichi Yamagishi, Christophe Veaux, and Kirsten MacDonald. 2019. CSTR VCTK corpus: English multi-speaker corpus for CSTR voice cloning toolkit (version 0.92). https://datashare.ed.ac.uk/handle/10283/3443

[62] Jixun Yao, Qing Wang, Yi Lei, Pengcheng Guo, Lei Xie, Namin Wang, and Jie Liu. 2023. Distinguishable Speaker Anonymization Based on Formant and Fundamental Frequency Scaling. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 1–5. doi:10.1109/ICASSP49357.2023.10095120

[63] Eric Zeng, Shrirang Mare, and Franziska Roesner. 2017. End User Security and Privacy Concerns with Smart Homes. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*. USENIX Association, Santa Clara, CA, 65–80. https://www.usenix.org/conference/soups2017/technical-sessions/presentation/zeng

[64] Weiyi Zhang, Shuning Zhao, Le Liu, Jianmin Li, Xingliang Cheng, Thomas Fang Zheng, and Xiaolin Hu. 2021. Attack on practical speaker verification system using universal adversarial perturbations. In *ICASSP 2021-2021 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2575–2579.

A Evaluation Results Tables

The following section includes tables that display a detailed breakdown of the evaluation metric results by dataset and speaker gender.

Table A1. Detailed results tables for the naive, informed, and adaptive EER evaluations. Scores are separated by dataset, speaker sex, and ASV model. Table A1a shows the EER_{naive} results, Table A1b shows the EER_{inf} results, and Table A1c shows the EER achieved during the adaptive attacker evaluation.

Dataset	Sex	No Obfuscation			McAdams			MicPro			VoiceMask			SafeSpeaker		
		ECAPA	X-Vec	ResNet	ECAPA	X-Vec	ResNet	ECAPA	X-Vec	ResNet	ECAPA	X-Vec	ResNet	ECAPA	X-Vec	ResNet
Libri_Test	M	8.76	9.85	0.55	33.21	35.40	16.94	29.94	29.20	19.19	11.68	15.29	4.01	39.87	45.66	46.55
	F	0.42	3.79	0.22	28.51	38.94	23.83	21.38	29.86	16.93	3.53	9.36	1.28	43.57	41.42	47.81
VCTK_Test	M	3.76	4.80	0.32	42.54	41.35	29.17	35.50	39.31	22.32	12.01	16.73	6.33	49.86	42.85	44.37
	F	2.10	3.05	0.09	36.59	34.93	18.46	36.41	30.49	14.98	12.92	9.21	1.96	46.51	49.62	45.85
Average		3.76	5.37	0.29	35.21	37.66	22.10	30.81	32.21	18.35	10.04	12.65	3.39	44.95	44.89	46.15

(a) Naive EER results broken down by dataset and speaker gender.

Dataset	Sex	No Obfuscation			McAdams			MicPro			VoiceMask			SafeSpeaker		
		ECAPA	X-Vec	ResNet	ECAPA	X-Vec	ResNet	ECAPA	X-Vec	ResNet	ECAPA	X-Vec	ResNet	ECAPA	X-Vec	ResNet
Libri_Test	M	8.76	9.85	0.55	32.12	29.20	22.63	18.43	22.79	18.98	10.95	13.50	2.01	12.27	20.94	28.01
	F	0.42	3.79	0.22	26.98	27.81	21.21	16.03	18.71	14.93	3.12	8.46	1.33	22.08	30.84	33.58
VCTK_Test	M	3.76	4.80	0.32	37.29	32.71	22.00	24.89	24.12	19.69	7.03	8.82	2.45	24.67	25.10	29.68
	F	2.10	3.05	0.09	30.01	26.33	15.70	17.99	20.28	14.60	6.06	6.54	1.72	26.72	30.79	34.20
Average		3.76	5.37	0.29	31.60	29.01	20.38	19.33	21.47	17.05	6.79	9.33	1.88	21.43	26.92	31.37

(b) Informed EER results broken down by dataset and speaker gender.

Dataset	Sex	No Obfuscation	McAdams	MicPro	VoiceMask	SafeSpeaker	SafeSpeaker-Rand
Libri_Test	M	8.76	5.84	11.13	8.58	1.38	30.29
	F	0.42	1.38	6.25	0.89	9.10	33.94
VCTK_Test	M	3.76	7.16	10.47	4.58	4.20	37.50
	F	2.10	4.07	5.82	2.15	6.68	39.00
Average		3.76	4.61	8.42	4.05	5.34	35.18

(c) Adaptive EER results broken down by dataset and speaker gender.

Table A2. Detailed results tables for the naive, informed, and adaptive TPR evaluations. Scores are separated by dataset, speaker sex, and ASV model. Table A2a shows the TPR_{naive} results, Table A2b shows the TPR_{inf} results, and Table A2c shows the TPR achieved during the adaptive attacker evaluation.

Dataset	Sex	No Obfuscation			McAdams			VoiceMask			MicPro			SafeSpeaker		
		1e-2	1e-3	1e-4	1e-2	1e-3	1e-4	1e-2	1e-3	1e-4	1e-2	1e-3	1e-4	1e-2	1e-3	1e-4
LibriSpeech	M	0.998	0.987	0.974	0.392	0.246	0.089	0.319	0.166	0.077	0.859	0.703	0.624	0.047	0.004	0.002
	F	1.000	0.998	0.989	0.405	0.296	0.225	0.323	0.078	0.009	0.984	0.947	0.815	0.036	0.007	0.000
VCTK	M	0.998	0.992	0.973	0.320	0.203	0.149	0.251	0.079	0.007	0.710	0.420	0.241	0.006	0.002	0.000
	F	1.000	1.000	0.999	0.478	0.332	0.223	0.335	0.126	0.035	0.960	0.836	0.650	0.032	0.004	0.000
Average		0.999	0.994	0.984	0.399	0.269	0.172	0.307	0.112	0.032	0.878	0.727	0.583	0.030	0.004	0.001

(a) Naive TPR results broken down by dataset and speaker gender.

Dataset	Sex	No Obfuscation			McAdams			VoiceMask			MicPro			SafeSpeaker		
		1e-2	1e-3	1e-4	1e-2	1e-3	1e-4	1e-2	1e-3	1e-4	1e-2	1e-3	1e-4	1e-2	1e-3	1e-4
LibriSpeech	M	0.998	0.987	0.974	0.374	0.181	0.082	0.458	0.266	0.131	0.965	0.887	0.659	0.523	0.176	0.073
	F	1.000	0.998	0.989	0.412	0.249	0.089	0.492	0.347	0.183	0.980	0.864	0.762	0.391	0.241	0.131
VCTK	M	0.998	0.992	0.973	0.365	0.201	0.127	0.366	0.166	0.061	0.948	0.827	0.710	0.242	0.088	0.018
	F	1.000	1.000	0.999	0.457	0.211	0.145	0.567	0.335	0.231	0.968	0.877	0.761	0.224	0.086	0.017
Average		0.999	0.994	0.984	0.402	0.211	0.111	0.471	0.279	0.152	0.965	0.864	0.723	0.345	0.178	0.060

(b) Informed TPR results broken down by dataset and speaker gender.

Dataset	Sex	No Obfuscation			McAdams			VoiceMask			MicPro			SafeSpeaker			SafeSpeaker-Rand		
		1e-2	1e-3	1e-4	1e-2	1e-3	1e-4	1e-2	1e-3	1e-4	1e-2	1e-3	1e-4	1e-2	1e-3	1e-4	1e-2	1e-3	1e-4
LibriSpeech	M	0.998	0.987	0.974	0.903	0.783	0.726	0.732	0.664	0.608	0.894	0.786	0.741	0.982	0.918	0.895	0.241	0.129	0.094
	F	1.000	0.998	0.989	0.978	0.927	0.849	0.851	0.695	0.577	0.993	0.969	0.929	0.805	0.714	0.651	0.224	0.104	0.064
VCTK	M	0.998	0.992	0.973	0.782	0.572	0.431	0.645	0.385	0.209	0.862	0.653	0.481	0.886	0.701	0.461	0.141	0.070	0.027
	F	1.000	1.000	0.999	0.899	0.765	0.640	0.863	0.725	0.544	0.953	0.877	0.796	0.821	0.614	0.455	0.128	0.050	0.006
Average		0.999	0.994	0.984	0.891	0.762	0.662	0.773	0.617	0.485	0.926	0.821	0.737	0.874	0.737	0.616	0.184	0.088	0.048

(c) Adaptive TPR results broken down by dataset and speaker gender.

Table A3. WER results broken down by dataset and speaker sex.

Dataset	No Obfuscation		McAdams		MicPro		VoiceMask		SafeSpeaker	
	Wav2Vec2	Whisper-Base	Wav2Vec2	Whisper-Base	Wav2Vec2	Whisper-Base	Wav2Vec2	Whisper-Base	Wav2Vec2	Whisper-Base
Libri_Test	1.844	11.064	9.999	12.905	26.368	21.235	2.503	10.245	6.937	18.201
VCTK_Test	4.643	16.304	22.493	23.035	59.801	41.549	9.370	17.906	20.373	25.575
Average	3.244	13.684	16.246	17.970	43.085	31.392	5.937	14.076	13.655	21.888

Received 1 November 2025; revised 1 May 2026; accepted 30 June 2026