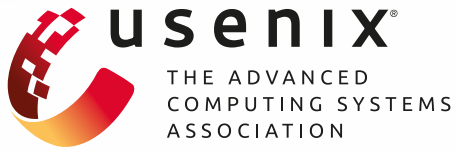

Void: A fast and light voice liveness detection system

Muhammad Ejaz Ahmed, *Data61, CSIRO;* Il-Youp Kwak, *Chung-Ang University;* Jun Ho Huh and Iljoo Kim, *Samsung Research;* Taekkyung Oh, *KAIST and Sungkyunkwan University;* Hyoungshick Kim, *Sungkyunkwan University*

https://www.usenix.org/conference/usenixsecurity20/presentation/ahmed-muhammad

This paper is included in the Proceedings of the 29th USENIX Security Symposium.

August 12–14, 2020

978-1-939133-17-5

Open access to the Proceedings of the 29th USENIX Security Symposium is sponsored by USENIX.

Void: A fast and light voice liveness detection system

Muhammad Ejaz Ahmed
Data61, CSIRO
Sungkyunkwan University

Il-Youp Kwak*
Chung-Ang University

Jun Ho Huh
Samsung Research

Iljoo Kim
Samsung Research

Taekkyung Oh
KAIST
Sungkyunkwan University

Hyoungshick Kim
Sungkyunkwan University

Abstract

Due to the open nature of voice assistants' input channels, adversaries could easily record people's use of voice commands, and replay them to spoof voice assistants. To mitigate such spoofing attacks, we present a highly efficient **vo**ice l**i**veness **d**etection solution called "Void." Void detects voice spoofing attacks using the differences in spectral power between live-human voices and voices replayed through speakers. In contrast to existing approaches that use multiple deep learning models, and thousands of features, Void uses a single classification model with just 97 features.

We used two datasets to evaluate its performance: (1) 255,173 voice samples generated with 120 participants, 15 playback devices and 12 recording devices, and (2) 18,030 publicly available voice samples generated with 42 participants, 26 playback devices and 25 recording devices. Void achieves equal error rate of 0.3% and 11.6% in detecting voice replay attacks for each dataset, respectively. Compared to a state of the art, deep learning-based solution that achieves 7.4% error rate in that public dataset, Void uses 153 times less memory and is about 8 times faster in detection. When combined with a Gaussian Mixture Model that uses Mel-frequency cepstral coefficients (MFCC) as classification features – MFCC is already being extracted and used as the main feature in speech recognition services – Void achieves 8.7% error rate on the public dataset. Moreover, Void is resilient against hidden voice command, inaudible voice command, voice synthesis, equalization manipulation attacks, and combining replay attacks with live-human voices achieving about 99.7%, 100%, 90.2%, 86.3%, and 98.2% detection rates for those attacks, respectively.

1 Introduction

Popular voice assistants like Siri (Apple), Alexa (Amazon) and Now (Google) allow people to use voice commands to quickly shop online, make phone calls, send messages, control smart home appliances, access banking services, and so on. However, such privacy- and security-critical commands make voice assistants lucrative targets for attackers to exploit. However, recent studies [11, 12, 23] demonstrated that voice assistants are vulnerable to various forms of voice presentation attacks including "voice replay attacks" (attackers simply record victims' use of voice assistants and replay them) and "voice synthesis attacks" (attackers train victims' voice biometric models and create new commands).

To distinguish between live-human voices and replayed voices, several voice liveness detection techniques have been proposed. Feng et al. [11] proposed the use of wearable devices, such as eyeglasses, or earbuds to detect voice liveness. They achieved about 97% detection rate but rely on the use additional hardware that users would have to buy, carry, and use. Deep learning-based approaches [7, 30] have also been proposed. The best known solution from an online replay attack detection competition called "2017 ASVspoof Challenge" [7] is highly accurate, achieving about 6.7% equal error rate (EER) – but it is computationally expensive and complex: two deep learning models (LCNN and CNN with RNN) and one SVM-based classification model were all used together to achieve high accuracy. The second best solution achieved 12.3% EER using an ensemble of 5 different classification models and multiple classification features: Constant Q Cepstral Coefficients (CQCC), Perceptual Linear Prediction (PLP), and Mel Frequency Cepstral Coefficients (MFCC) features were all used. CQCC alone is heavy and would consist of about 14,000 features.

To reduce computational burden and maintain high detection accuracy, we present "Void" (**Vo**ice l**i**veness **d**etection), which is a highly efficient voice liveness detection system that relies on the analysis of cumulative power patterns in spectrograms to detect replayed voices. Void uses a single classification model with just 97 spectrogram features. In particular, Void exploits the following two distinguishing characteristics in power patterns: (1) Most loudspeakers *inherently* add distortions to original sounds while replaying them. In

*Part of this work done while Dr. Kwak was at Samsung Research.

consequence, the overall power distribution over the audible frequency range often show some uniformity and linearity. (2) With human voices, the sum of power observed across lower frequencies is relatively higher than the sum observed across higher frequencies [15, 29]. As a result, there are significant differences in the cumulative power distributions between live-human voices and those replayed through loudspeakers. Void extracts those differences as classification features to accurately detect replay attacks.

Our key contributions are summarized below:

- Design of a *fast* and *light* voice replay attack detection system that uses a single classification model and just 97 classification features related to signal frequencies and cumulative power distribution characteristics. Unlike existing approaches that rely on multiple deep learning models and do not provide much insight into complex spectral features being extracted [7, 30], we explain the characteristics of key spectral power features, and why those features are effective in detecting voice spoofing attacks.
- Evaluation of voice replay attack detection accuracy using two large datasets consisting of 255,173 voice samples collected from 120 participants, 15 playback devices and 12 recording devices, and 18,030 ASVspoof competition voice samples collected from 42 participants, 26 playback speakers and 25 recording devices, respectively, demonstrating 0.3% and 11.6% EER. Based on the latter EER, Void would be ranked as the second best solution in the ASVspoof 2017 competition. Compared to the best-performing solution from that competition, Void is about 8 times faster and uses 153 times less memory in detection. Void achieves 8.7% EER on the ASVspoof dataset when combined with an MFCC-based model – MFCC is already available through speech recognition services, and would not require additional computation.
- Evaluation of Void's performance against hidden command, inaudible voice command, voice synthesis, equalization (EQ) manipulation attacks, and combining replay attacks with live-human voices showing 99.7%, 100%, 90.2%, 86.3%, and 98.2% detection rates, respectively.

2 Threat Model

2.1 Voice replay attacks

We define live-human audio sample as a voice utterance initiated from a human user that is directly recorded through a microphone (such that would normally be processed by a voice assistant). In a voice replay attack, an attacker uses a recording device (e.g., a smartphone) in a close proximity to a victim, and first records the victim's utterances (spoken words) of voice commands used to interact with voice assistants [3, 11, 12]. The attacker then replays the recorded samples using an in-built speaker (e.g., available on her phone) or a standalone speaker to complete the attack (see Figure 1).

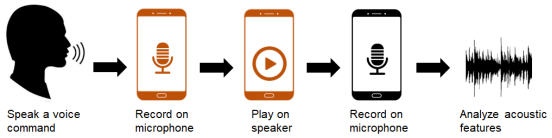

Figure 1: Steps for a voice replay attack.

Voice replay attack may be the easiest attack to perform but it is the most difficult one to detect as the recorded voices have similar characteristics compared to the victim's live voices. In fact, most of the existing voice biometric-based authentication (human speaker verification) systems (e.g., [31, 32]) are vulnerable to this kind of replay attack.

2.2 Adversarial attacks

We also consider more sophisticated attacks such as "hidden voice command" [24, 25], "inaudible voice command" [18–20], and "voice synthesis" [6, 12] attacks that have been discussed in recent literature. Further, EQ manipulation attacks are specifically designed to game the classification features used by Void by adjusting specific frequency bands of attack voice signals.

3 Requirements

3.1 Latency and model size requirements

Our conversations with several speech recognition engineers at a large IT company (that run their own voice assistant services with millions of subscribed users) revealed that there are strict latency and computational power usage requirements that must be considered upon deploying any kind of machine learning-based services. This is because additional use of computational power and memory through continuous invocation of machine learning algorithms may incur (1) unacceptable costs for businesses, and (2) unacceptable latency (delays) for processing voice commands. Upon receiving a voice command, voice assistants are required to respond immediately without any noticeable delay. Hence, processing delays should be close to 0 second – typically, engineers do not consider solutions that add 100 or more milliseconds of delay as portable solutions. A single GPU may be expected to concurrently process 100 or more voice sessions (streaming commands), indicating that machine learning algorithms must be *lightweight*, *simple*, and *fast*.

Further, as part of future solutions, businesses are considering on-device voice assistant implementations (that would not communicate with remote servers) to improve response latency, save server costs, and minimize privacy issues related to sharing users' private voice data with remote servers. For such on-device solutions with limited computing resources available, the model and feature complexity and size (CPU

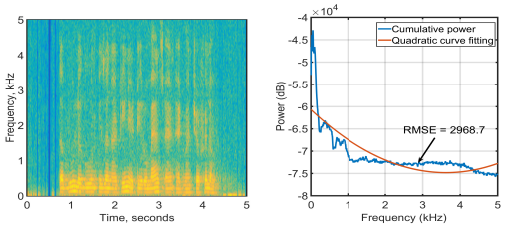

Figure 2: Spectrogram of an example phrase *"The Blue Lagoon is a 1980 romance and adventure film"* lively uttered by a human user (left), and cumulative power spectral decay of the corresponding command (right).

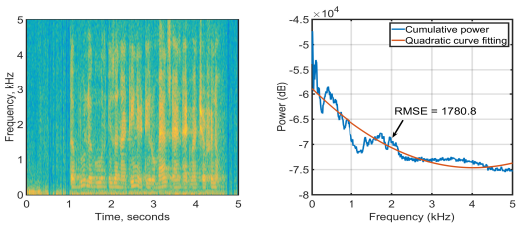

Figure 3: Spectrogram of the same example phrase (as in Figure 2) replayed using iPhone 6S Plus (left), and cumulative power spectral decay (right).

and memory usage) requirements would be even more constraining.

3.2 Detection accuracy requirements

Our main objective is to achieve competitively high accuracy while keeping the latency and resource usage requirements at acceptable levels (see above). Again, our conversations with the speech recognition engineers revealed that businesses require around 10% or below EER to be considered as a usable solution. For reference, the best performing solution from the ASVspoof 2017 competition achieved 6.7% EER [30], and the second best solution achieved 12.3% [7].

4 Key classification features

Void exploits the differences in frequency-dependent spectral power characteristics between live-human voices and voices replayed through loudspeakers. Through numerous trials and experiments, we observed three distinct features related to power spectrum of speech signals that may distinguish live-human voices from voices replayed through loudspeakers. This section explores those features in detail.

Figure 1 shows the steps involved in replaying recorded voice signals. An attacker would first record a victim's voice command using her own recording device. Then the attacker would use the same device (in-built speaker) to replay the recorded voice command, targeted at the victim's device. This attack command is then processed by the voice assistant service running on the victim's device. While performing this replay attack, some distortions may be added to the victim's original sound while being recorded with the microphone on the attacker's device, and also while being replayed through the in-built speaker due to hardware imperfections. The following sections explore the spectral power characteristics of replayed voices, and analyze key classification features that are used to classify voice replay attacks.

4.1 Decay patterns in spectral power

In general, low quality loudspeakers are designed to achieve high sensitivity and volume but at the cost of compromising audio fidelity and adding unwanted distortions [35]. As a result, distortions that contribute to non-linearity may be more prevalent in low quality loudspeakers, and less visible in high quality loudspeakers [36, 37].

Figure 2 (left) shows the spectrogram of a sentence *"The Blue Lagoon is a 1980 romance and adventure film"* uttered live, and processed by an audio chipset in a laptop. Here, the audio sampling rate was 44.1kHz, and the utterance duration was 5 seconds. In this voice sample, most of the spectral power lies in the frequency range between 20Hz and 1kHz. The cumulative spectral power measured for each frequency is also shown in Figure 2 (right). There is an *exponential power decay* of human voice at frequency around 1kHz.

On the other hand, the spectrogram of a phrase replayed through iPhone 6s Plus in-built speaker (see Figure 3) shows some *uniformity – spectrum spread* is shown in the power distributions between 1 and 5kHz. Unlike live-human voice trends shown in Figure 2, the cumulative spectral power does not decrease exponentially; rather, there is a relatively more linear decay between 1 and 5kHz. To show the difference between Figure 2 and 3 quantitatively, we added quadratic fitting curves on them and computed Root Mean Square Error (RMSE) separately.

Our experimentation with 11 in-built smartphone speakers showed similar behaviors in their spectral power distributions; i.e., power decreased gradually across frequencies and did not decay exponentially. An example cumulative distribution of spectral power density is shown in Figure 4. With the human voice example, about 70% of the overall power lies in the frequency range below 1kHz. However, in the loudspeaker case, the cumulative distribution increases almost linearly, and 70% of the total power lies within the frequency range of about 4kHz.

One possible explanation for this spreading out characteristic is low-quality hardware boosting power in certain frequency ranges. Consequently, such a linear decay pattern in spectral power (over audible frequency range) could be

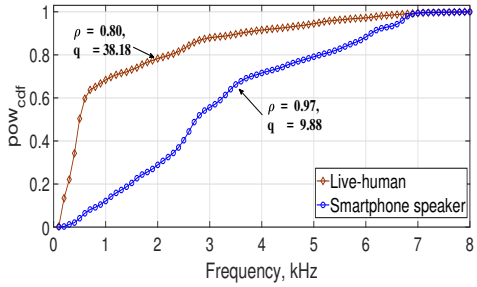

Figure 4: Cumulative distribution of spectral power density over frequencies, showing up to 8kHz ($W = 10$).

trained and used to classify voices replayed through low-quality loudspeakers. Appendix A demonstrates that three signal power features would be used to classify live-human voices and voices replayed through 11 in-built smartphone speakers.

4.2 Peak patterns in spectral power

Because high-quality standalone loudspeakers boost power across a wide range of frequencies to reduce non-linear distortions, the linear decay patterns described above may not be sufficient against such loudspeakers.

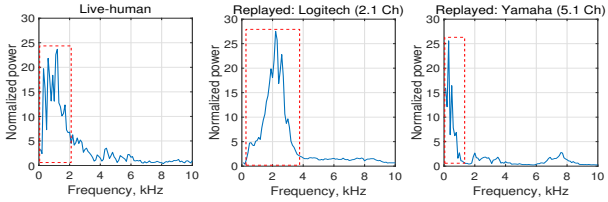

Figure 5: Signal power frequency range between 20Hz and 10kHz of the spectrogram of the same example phrase (as in Figure 2). Live-human voice (left): fine-grained power fluctuations can be observed over the frequency range from 20Hz to 2kHz. High-quality speakers (middle and right): the power over the same frequency range is more concentrated with less fluctuations.

Figure 5 compares normalized signal power of live-human voices and voices replayed through two different high-quality loudspeakers. Even though they show similar exponential decay patterns overall, the low frequency patterns are different (see red-dashed rectangles in Figure 5). As for loudspeakers (middle and right), there is a smaller number of sharp and long peaks at low frequencies compared to live-human voices (left).

Thus, distortion-induced power patterns (e.g., the number of visible power peaks, their corresponding frequencies, and standard deviations of power peaks sizes) in low frequencies could be effective in detecting standalone speakers that produce higher quality sounds. We also use higher order polynomials to accurately model spectral power shapes, and use these models to identify more fine-grained differences in spectral power patterns between live-human and replayed samples (see Figure 5). We also provide power patterns for different loudspeakers in Appendix B.

4.3 Linear prediction cepstrum coefficients (LPCC)

Because the decay and peak patterns discussed in Sections 4.1 and 4.2 mainly look at specific frequency ranges. To perform a more general inspection of wider frequency ranges, we additionally use linear prediction cepstrum coefficients (LPCC) [4] as a *complementary* feature.

LPCC is popularly used for auditory modeling in speech-related applications. The key idea behind LPCC is that a speech sample can be approximated as a linear combination of previous samples. LPCC for a voice sample is computed by minimizing the sum of squared differences between the voice sample and linearly predicted ones. The computational complexity of LPCC is lower than MFCC since LPCC does not require computation of discrete Fourier transform [5]. We chose LPCC as a complementary, lightweight feature to help Void utilize spectral features covering wider frequency ranges of speech signals.

5 System design

We designed Void to satisfy the requirements specified in Section 3 based on the key classification features described in Section 4. To detect replay attacks, Void analyzes signal power distributions over the audible frequency range – computing linearity degree of given signal power, and identifying peak patterns in low-power and high-power frequencies.

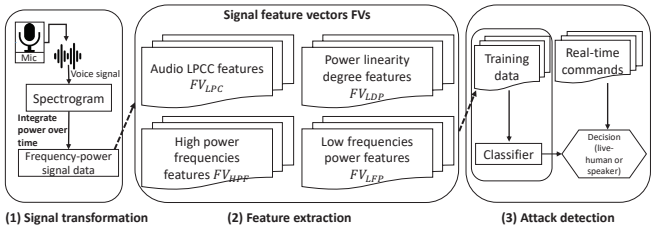

Figure 6: High-level design of Void.

5.1 Void overview

Attack detection through Void involves three stages as shown in Figure 6: signal transformation, feature extraction, and real-time attack detection. The overall Void algorithm is described in Algorithm 1. A voice command $Voice_{\text{in}}$, window size W, and a weighting factor ω are given as inputs to Algorithm 1.

S_{pow} represents the cumulative spectral power per frequency of $Voice_{\text{in}}$. W represents the size of a single segment of S_{pow} to fully capture the dynamic characteristics of S_{pow} with a small number of segments. A weighting factor ω between 0 and 1 is used to calculate a threshold for feature values in higher frequencies. Those parameter values were determined experimentally with a large number of training samples. Last, $pow(i)$ represents the accumulated power in i_{th} segment of S_{pow}. We only consider voice signals below 15kHz because most of the signal power for voice samples fall below 15kHz.

Algorithm 1 Void's overall procedure.

Input: $Voice_{\text{in}}$, W and ω
Output: `live-human` or `replayed`
Stage 1: Signal transformation
1: Compute STFT of for input voice command $Voice_{\text{in}}$
2: Compute S_{pow} from STFT
Stage 2: Feature extraction
3: Divide S_{pow} into k segments where $k = \lfloor \frac{\text{size}(S_{pow})}{W} \rfloor$.
4: **for** i_{th} segment Seg_i from $i = 1$ to k **do**
5: $pow(i)$ = the sum of power in Seg_i.
6: $< pow >$ = Vectorize($pow(1), \cdots, pow(k)$) and normalize between 0 and 1
7: FV_{LFP} = First 48 values of $< pow >$
8: FV_{LDF} = LinearityDegreeFeatures($< pow >$)
9: FV_{HPF} = HighPowerFrequencyFeatures(FV_{LFP}, ω)
10: Compute LPCC of $Voice_{\text{in}}$ and store the results as FV_{LPC}
Stage 3: Attack detection
11: $FV_{\text{Void}} = \{FV_{LDF}, FV_{HPF}, FV_{LPC}, \text{and } FV_{LFP}\}$
12: Run SVM classifier with FV_{Void} and provide the class label (either `live-human` or `replayed`) as output

5.2 Signal transformation

In the first signal transformation stage, given an input voice signal $Voice_{\text{in}}$, short-time Fourier transform (STFT) is computed (Step 1 of Algorithm 1). To compute STFT, a given voice signal is divided into short chunks of equal length (denoted as $wlen$ = 1,024); Fourier transform is then computed on each chunk. We used a periodic Hamming window length $wlen$ of 1,024, and a hop length of 256, which is computed by $wlen/4$. The number of fast Fourier transform points used ($nfft$) for computing STFT is set to 4,096. The obtained signal spectrogram contains frequencies and corresponding power over time (see Figure 2 (left)). From the computed STFT, cumulative spectral power per frequency (S_{pow}) is computed (Step 2 of Algorithm 1). The terms "cumulative spectral power" and "power" are used hereafter interchangeably. S_{pow} is a vector that contains the total accumulated power for each frequency over the full duration of $Voice_{\text{in}}$ (see Figure 2 (right)). S_{pow} obtained from STFT is a vector of size 1,500 (Step 2 of Algorithm 1). We use the notation $size(S_{pow})$ to represent the number of values stored in S_{pow}.

5.3 Feature extraction

The vector S_{pow} computed from the first stage is used as the input to the second stage to extract the classification features. Void sequentially computes the following four types of features: (1) low frequencies power features (FV_{LFP}), (2) signal power linearity degree features (FV_{LDF}), (3) higher power frequencies features (FV_{HPF}), and (4) LPCC features for audio signals (FV_{LPC}). FV stands for feature vectors. The first three feature classes are computed from S_{pow} while FV_{LPC} is computed directly from raw voice signals $Voice_{\text{in}}$.

5.3.1 Low frequencies power features

In the second stage of Algorithm 1, we first divide the signal S_{pow} into k short segments of equal-length according to the given window size W (see Step 3). We empirically set $W = 10$. If the size of S_{pow} can not be divided by W, we simply omit the last segment. Next, we compute the sum of power in each segment Seg_i for $i = 1$ to k (see Steps 4 and 5). We then vectorize the first k segments of power density values as $< pow > (= pow(1), \ldots, pow(k))$ (see Step 6). The vector $< pow >$ is directly used in FV_{LFP} (see Step 7). After executing this step, we would have cumulative spectral power density values for all k segments. Power density values for each segment are ordered by frequency, starting from the lowest frequency of a given voice sample. We are only interested in retaining power density values within the frequency value of 5kHz because our experiments showed that there are clear differences between human and replayed voices at the lower frequencies below 5kHz (see Figure 5). Therefore, we keep just the first 48 values of $< pow >$ vector and assign them to FV_{LFP} (see Step 7).

5.3.2 Signal power linearity degree features

Given the vector $< pow >$ of k segments, we compute the signal's feature vector (FV_{LDF}) to measure the degree of linearity (as discussed in Section 4.1).

Algorithm 2 LinearityDegreeFeatures

Input: $< pow >$
Output: FV_{LDF}={ρ, q}.
1: Normalize $< pow >$ with $sum(< pow >)$ to obtain $< pow >_{\text{normal}}$
2: Accumulate the values of $< pow >_{\text{normal}}$ to obtain pow_{cdf}
3: Compute the correlation coefficients of pow_{cdf} and store the results as ρ
4: Compute the quadratic coefficients of pow_{cdf} and store the results as q

Algorithm 2 describes the procedure for computing the linearity degree of $< pow >$. Initially, $< pow >$ is normalized by dividing each value in $< pow >$ by the total signal power ($sum(< pow >)$) (see Step 1 in Algorithm 2). The normalized power signal vector $< pow >_{\text{normal}}$ is then used to compute the cumulative distribution of spectral power, denoted by pow_{cdf} (see Step 2). In this step, $< pow >_{\text{normal}}$ is accumulated in a step-wise fashion.

For the linearity degree of pow_{cdf}, we compute the following two features (see Step 3 and 4): correlation coefficients ρ and quadratic curve fitting coefficients q of pow_{cdf} (see

Appendix C). Correlation coefficients of a cumulative distribution can be used to quantify the linearity of the cumulative distribution. However, we found that ρ is not highly sensitive in identifying the distinguishable exponential growth of power in live-human voices at frequencies between 20Hz and 1kHz (see Figure 5). Therefore, we introduce the quadratic curve fitting coefficients q of signal pow_{cdf} as another metric to quantify the degree of linearity for the cumulative distribution function. Finally, the two computed coefficients $\{\rho, q\}$ are stored as FV_{LDF}.

5.3.3 High power frequency features

Given the vector $< pow >$ and the peak selection threshold ω, we compute the feature vector (FV_{HPF}) to capture the dynamic characteristics of spectral power (see Appendix D).

Algorithm 3 HighPowerFrequencyFeatures

Input: FV_{LFP} and ω
Output: $FV_{HPF} = \{N_{peak}, \mu_{peaks}, \sigma_{peaks}, P_{est}\}$
1: Find peaks from FV_{LFP} and store the discovered peaks $\{(peak_1, loc_1), \cdots, (peak_n, loc_n)\}$ as S_{peak} ▷ n is the number of peaks discovered in FV_{LFP}
2: $T_{peak} = \omega \cdot max(peak_1, \cdots, peak_n)$
3: **for** each $peak_i$ in S_{peak} from $i = 1$ to n **do**
4: **if** $peak_i < T_{peak}$ **then** remove $peak_i$ from S_{peak}
5: N_{peak} = the number of peaks in S_{peak};
6: μ_{peak} = the mean of the locations of peaks in S_{peak}
7: σ_{peak} = the standard deviation of the locations of peaks in S_{peak}
8: P_{est} = estimated coefficients to fit a polynomial of order 6 to FV_{LFP}

Algorithm 3 describes the procedure for computing high power frequency features (FV_{HPF}). In $< pow >$, we first identify peaks and their locations (see Step 1). Our peak selection criterion T_{peak} automatically scales itself with respect to the spectral power density values of a given signal. For example, for a given low or high power voice signal, T_{peak} is computed accordingly as shown in Step 2. We experimentally found that detected peaks from live-human voice samples and replayed samples show different characteristics when we set $\omega = 0.6$. However, ω needs to be configured such that the high power frequency features are effective in detecting replayed voices. We set a threshold value to filter out insignificant peaks, multiplying $max(S_{peak})$ by a given weighting factor ω where $0 \leq \omega \leq 1$ (see Step 2, 3, and 4).

To construct FV_{HPF}, we first count the number of peaks in S_{peak} and store the number of counted peaks as N_{peak} (see Step 5); the mean and standard deviation of locations of the discovered peaks are sequentially computed and stored as μ_{peaks} and σ_{peaks}, respectively (see Step 6 and 7); and we determine the 6 order of the polynomial to be fitted to FV_{LFP} and use the polynomial coefficients as P_{est} (see Step 8).

5.3.4 LPCC features

We use an auto-correlation method with Levinson-Durbin algorithm [26] to compute LPCC for a given speech signal, generating 12 coefficients. These 12 LPCC coefficients are stored in the feature vector FV_{LPC}.

5.4 Attack detection

In the third stage of Algorithm 1, we construct a classifier with all the feature sets computed in Section 5.3 to detect attacks performed through loudspeakers. Instead of manually constructing detection rules, we opted to utilize machine learning-based classifiers as follows:

Feature set. The four feature vectors FV_{LFP}, FV_{LDF}, FV_{HPF}, and FV_{LPC} are combined as a feature set for classification algorithm. The total number of features is 97. We used Classification and Regression Trees (CART) [28] to analyze the relative importance of individual classification features. Figure 7 shows the importance scores computed for individual features based on the classifications performed on the ASVspoof dataset.

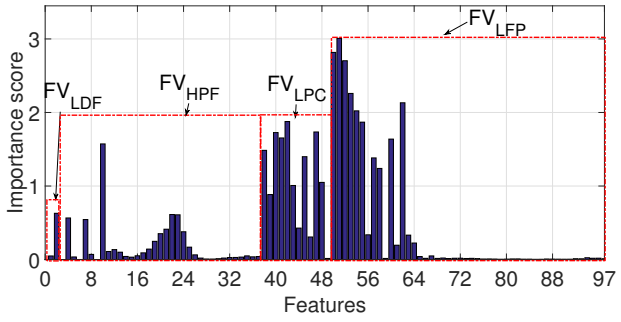

Figure 7: Importance scores for individual features on the ASVspoof dataset.

There were 17 (with the scores above 1.0) noticeably important features (shown by the peaks) from the 4 feature groups visualized in red-dashed rectangles. We can observe that FV_{LFP} and FV_{LPC} are the most important features. From the FV_{HPF} group, we found that P_{est} features play an important role in distinguishing voice replay attack. However, some power value features in the low frequencies group (FV_{LFP}) were relatively less important. To show the necessity of all features used in Void, we also tested Void separately on each of the feature groups: FV_{LFP}, FV_{LDF}, FV_{HPF}, and FV_{LPC} (see Appendix E).

Classifier. To implement a lightweight system, we need to build a classifier based on the four feature vectors, which achieves high detection accuracy and meets the latency requirements. Our recommended classification algorithm is described in Section 7.2. We also provide the details of Void's implementation parameters (see Appendix F).

6 Data Collection

This section describes human voice samples and voice attack samples we collected using multiple recording and playback devices, and under varying conditions. For our own dataset, all of the voice samples were recorded at a sampling frequency (F_s) of 44.1kHz. We also used a publicly available replay attack dataset that was used in the 2017 voice spoofing attack detection (ASVspoof) competition [7]. The ASVspoof dataset evaluation results were used to directly compare Void's performance against the top performing (state of the art) solutions from the competition.

6.1 Demographics and human voice collection

We recruited a total of 120 participants from two different locations (a university and a company), and asked each participant to say around 50 commands from a prepared list of real-world voice assistant commands. We used two different smartphones, Galaxy S8 and iPhone 8 to record all human voices. After eliminating voice samples that were not recorded properly or were not understood by the voice assistant, we were left with 10,209 human voice samples to experiment with. The voice commands were mixed in lengths (approximately ranging from 2 to 5 seconds) and command types (e.g., setting alarms, calling contacts, and opening emails). About 53% of the participants were male, ensuring that both male and female voice frequency ranges were covered [16]. Most of the participants were in the 40-49 (13%), 30-39 (62%), and 20-29 (25%) age groups.

We explicitly informed the participants that the purpose of the voice sample collection was to develop and evaluate a voice liveness detection solution. Ethical perspective of our research was validated through an institutional review board (IRB) at Sungkyunkwan university; the IRB file number is "2018-01-024."

6.2 Replay attack dataset

To generate a comprehensive replay attack dataset, we replayed all 10,209 human voice samples in an open lab environment through a mixed set of speakers and recorded them under varying conditions as described below:

- **Background noise:** The open lab environment we used to record all attack samples is collaboratively used by about 100 employees at a large IT company. During the day, the lab is used for discussions, development, testing, coffee breaks, and so on. The lab is quiet in the evenings. There are also daily background noises generated from server machines, TVs, projectors, and robot cleaners. The human voices were replayed and recorded throughout the day and in the evenings, capturing natural yet diverse set of background noises as well as silent moments while generating the replay attack dataset.
- **Distances between attacking devices and target devices:** Distances between target devices (used to record voice samples) and attack devices (used to play recorded voice samples) could affect the detection rate because spectral power features could be affected with distance. Hence, we recorded replayed voice samples in three varying distances: about 15 centimeters, 130 centimeters, and 260 centimeters away from each playback speaker.
- **Playback speaker types:** We used 11 different types of in-built speakers including smartphones and a smart TV, and four different types of standalone speakers to replay recorded voice samples (see Appendix G). Each standalone speaker was different in terms of the number of sound channels supported, brand, price, and electrical power. Our standalone speaker selection included Logitech 2.1 ch., and Yamaha 5.1 ch. speakers that were designed to optimize the final acoustic sounds for human ears. We replayed about 5,500 human voices through each speaker type. Only the Yamaha 5.1 channel speaker was connected to the replaying devices (smartphones) through Bluetooth. The other three standalone speakers were all connected through auxiliary port (AUX) physical cables.
- **Recording device types (microphones):** We used 3 different laptops, and 9 different smartphones as recording devices (see Appendix H). For each playback speaker type, we used a different combination of three recording devices with varying distances as described above.

After eliminating voice samples that were not recognized properly by voice assistants, we were left with a final attack set of 244,964 samples to experiment with. All voice samples were recorded, stored, and analyzed in the "WAV" file format. The details of the dataset are presented in Table 1.

Table 1: Replay attack dataset description.

	Detail	Our dataset	ASVspoof
# Data	Live-human	10,209	3,565
	Attack	244,964	14,465
	Participants	120	42
# Devices	Speakers	15	26
	Recording mics	12	25
# Configurations		33	125

6.3 ASVspoof 2017 dataset

We also evaluated Void's performance against an online replay attack database referred to as the "2017 ASVspoof Challenge dataset," which was created to facilitate an online competition for detecting voice spoofing attacks [8]. The entire dataset (all three sets combined) contains voice samples collected through 177 replay attack sessions, where each session consists of voice samples that were recorded under varying replay

configurations, and at a sampling frequency of 16kHz. Each replay configuration is different with respect to recording device type, playback device type, and recording environment. Recording environments include balconies, bedrooms, canteens, homes, and offices. 26 playback devices were used, including 12 high-quality professional audio equipment such as active studio monitors and studio headphones (e.g., Genelec 8020C and Behringer Truth B2030A). Such devices would introduce much less acoustic distortion than smaller, in-built loudspeakers. Nine playback devices were in-built speakers from various smartphones, tablets, and laptops. 5 devices were medium-quality, portable speakers (e.g., Creative A60 speakers). 25 recording devices were used, including 12 high-quality recording devices such as studio-quality condenser microphones or hand-held recorders (e.g., Zoom H6 recorder with Behringer ECM8000 mic). There were 61 replay configurations used.

The ASVspoof dataset is partitioned into training set, development set, and evaluation set (see Table 2). We trained Void on the training and development sets, and tested Void's performance against the evaluation set, which is compliant with the ASVspoof competition rules (see [9]).

Table 2: Description of ASVspoof 2017 dataset [8].

Partition	# Speakers	Live-human	Replay
Training	10	1,507	1,507
Development	8	760	950
Evaluation	24	1,298	12,008
Total	42	3,565	14,465

The training set and developing set combined consists of 2,267 live-human samples and 2,457 attack samples. The evaluation set consists of 1,298 live-human samples and 12,008 attack samples – this proportion of attack samples in the evaluation set is much larger (see Table 1).

7 Evaluation

7.1 Experiment setup

For evaluation, we used the two datasets described in Section 6. As for the first attack dataset that we collected, to reduce any bias that might be associated with the process of randomly splitting the datasets into training and testing sets, we used 10 fold cross-validation: the training samples were partitioned into 10 equal-sized sets with similar class distributions. As for the ASVspoof dataset, we trained Void using both the train and developing sets, and evaluated Void against the evaluation set – this is how the competition measured the performance of submitted solutions.

To measure the performance of Void, we rely on the standard speaker verification metrics, which are "false acceptance rates" (FAR) and "false rejection rates" (FRR). The four possible classification decisions are presented in Table 3. "True acceptance" (TA) and "true rejection" (TR) refer to correctly detecting live-human voice and loudspeaker, respectively. "False acceptance" (FA) is when a loudspeaker is mis-classified as live-human voice, and "false rejection" (FR) is when live-human voice is mis-classified as loudspeaker. We measure equal error rates (EERs), representing error rates for which FAR and FRR are equal. Receiver operating characteristic (ROC) curve and area under the curve (AUC) were also used for comparison of various thresholds. For computing EER, we used the Bosaris toolkit (https://sites.google.com/site/bosaristoolkit/) that was suggested in the 2017 ASVspoof competition [7].

Table 3: Four possible classification decisions.

	Accept	Reject
Live-human	True Acceptance	False Rejection
Replay attack	False Acceptance	True Rejection

Our experiments were conducted on a powerful server equipped with two Intel Xeon E5 (2.10GHz) CPUs, 260GB RAM and NVIDIA 1080Ti GPU, running 64-bit Ubuntu 16.04 LTS operating system. Our latency and model complexity results were measured based on this server setting.

7.2 Optimal classification method for Void

To determine the optimal classification method, we first evaluated the performance of five different classification methods that are popularly used in security systems: k-Nearest Neighbor (kNN), Random forest, SVM with linear kernel (SVM linear), and SVM with RBF kernel (SVM RBF). All of those classifiers were tested using the ASVspoof dataset.

Table 4 shows the detection accuracy of four classification models (k-Nearest Neighbor (kNN), Random forest, SVM with linear kernel (SVM linear), and SVM with RBF kernel (SVM RBF)) for the ASVspoof dataset.

Table 4: Detection accuracy of four classification algorithms for the ASVspoof dataset [7].

Algorithm	EER (%)
SVM RBF	11.6
Random forest	23.4
SVM linear	15.8
kNN	19.1

Among classification algorithms tested, SVM RBF produced the best EER results (11.6%) while providing training and testing times comparable with other classification algorithms. Therefore, we recommend the use of SVM RBF. All

subsequent evaluations were conducted using the SVM RBF classifier.

7.3 Attack detection accuracy

We show the ROC curve and AUC in Figure 8 to demonstrate the classification performance of Void under various threshold settings. Void achieved an AUC of 0.99 and 0.94 for our dataset and ASVspoof dataset, respectively. Even though the live-human to replay attack sample ratios are low in both datasets, the strong ROC curve and AUC results indicate that Void would likely achieve low error rates when more balanced datasets are used (see Figure 8). Void achieved an EER of 0.3% and 11.6% for our dataset and ASVspoof dataset, respectively[1]. We note that this EER result (11.6%) would rank Void as the second best solution (EER 12.34%) in the ASVspoof 2017 competition [10].

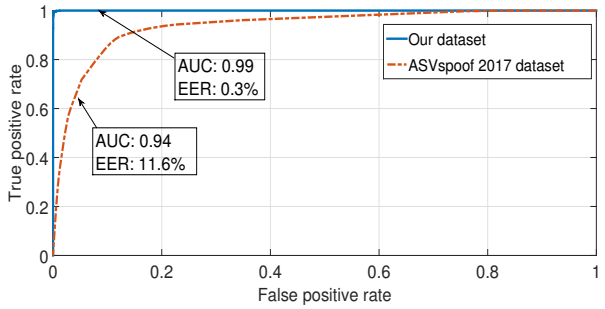

Figure 8: Accuracy results of Void.

To compare Void against existing solutions from the ASVspoof competition with respect to latency, space complexity, and accuracy, we implemented (used existing code if available) the two classification models described below, and evaluated them using the ASVspoof evaluation set. Table 5 summarizes those evaluation results.

Table 5: Average training/testing times, number of features used, average memory used, and performance of classification models on the ASVspoof dataset [7].

	Measure	Void	CQCC-GMM [7]	STFT-LCNN [30]
Time	Extraction (sec.)	0.035	0.059	$3e^{-4}$
	Training (sec.)	0.283	6,599.428	15,362.448
	Testing (sec.)	0.035	0.062	0.270
Memory	# Features	97	14,020	84,770
	Memory size (MB)	1.988	173.707	304.176
Accuracy	EER	11.6%	23.0%	7.4%

CQCC-GMM. This is the baseline approach recommended in the 2017 ASVspoof competition [7] that uses CQCC as the main features, and 512-component Gaussian Mixture Model (GMM) as the classification algorithm. CQCC-GMM achieved 23% EER on the ASVspoof evaluation set [7] – demonstrating significantly larger EER compared to Void.

STFT-LCNN. To evaluate the best performing model from the ASVspoof competition, we implemented the Light Convolutional Neural Network (LCNN) structure described in [30] and used STFT as the main features – this is one of the two deep learning models used. We contacted the authors from [30] and used the exact LCNN hyper-parameters and STFT parameters they recommended. Their model consists of 5 convolutional layers, 4 network in network layers, 10 max-feature-map layers, 4 max-pooling layers, and 2 fully connected layers as described in [30]. STFT-LCNN achieved 7.4% EER on the ASVspoof evaluation set [7] according to the EER result presented in [30][2].

7.4 Latency and model complexity results

We compare Void against CQCC-GMM and STFT-LCNN with respect to the latency and model complexity (see Table 5). Feature extraction time ("Extraction") represents the average time taken to extract features from a single voice sample. Training time ("Training") refers to the time taken to train a model (using the extracted features). Testing time refers to the average time taken to extract features from a single voice sample and perform classification using those features. Memory size, in megabytes, refers to the average memory used by each model to classify a given sample.

As for the space complexity, we count the number of features extracted from a single voice sample. The number of features used by Void is just 97, compared to 14,020 features used by our CQCC-GMM implementation and 84,770 features used by STFT-LCNN. In consequence, Void only used 1.988 megabytes of memory on average to classify a given voice sample. CQCC-GMM used 173.707 megabytes and STFT-LCNN used 304.176 megabytes of memory.

As for the execution time overheads, on average, Void took 0.283 seconds for training, and 0.035 seconds for testing. Void outperformed all other solutions with respect to both the training time and testing time. The average testing time for STFT-LCNN was 0.27 seconds.

These observations clearly indicate that Void is a much more efficient, faster, and lighter solution compared to other solutions. Void is the only solution that would satisfy the strict latency, and model and feature complexity requirements described in Section 3.1.

[1] We additionally provide precision, recall, and F1-score measures for our dataset because the numbers for live-human samples and attack samples are imbalanced; the precision, recall, and F1-score are 95.8%, 85.2%, and 89.2%, respectively.

[2] Although we used the same hyper-parameters and model layouts descried in [30], our own implementation achieved 12.7% EER – higher than the 7.4% EER presented in [30].

7.5 Using Void as an ensemble solution

Our discussions with several speech recognition engineers at a large IT company revealed that filter bank and MFCC are the only two spectral features used for speech recognition. Since MFCC would be extracted and available anyway (and would not require any additional feature extraction time), we implemented an ensemble solution that consists of MFCC-GMM and Void, and evaluated its accuracy against the ASVspoof evaluation set. MFCC-GMM alone achieves 25.5% EER on the evaluation set, and uses 8,053 features – it is much lighter than CQCC. Its average testing time was around 0.03 seconds.

We used a logistic regression model to compute the optimal weight for each model: 0.7 for Void, and 0.3 for MFCC-GMM. This ensemble approach achieved an EER of 8.7%, further demonstrating the effectiveness of Void and its potential benefits when combined with other lightweight models. Again, our ensemble solution would have ranked second in the ASVspoof competition, and not too far from the best solution that achieved an EER of 6.74%. The total testing time would be around 0.06 seconds per voice sample.

7.6 Effects of variances

In this section, we analyze the effects of four key variances – distances between target devices and attack devices, human gender, loudspeaker types and cross data training – on the performance of Void. We trained a single classifier using our own dataset; the train set comprised of 9,000 live-human samples and 9,000 replay attack samples. We used this classifier to evaluate Void's performance under distance and gender variances.

7.6.1 Attack source distances

To analyze the effects of varying distances between attacker and target device, voice samples were collected using three different distances: 15cm, 130cm, and 260cm. For testing, we used the remaining replayed samples, randomly choosing 1,920, 1,919, 1,920 samples, respectively, from each of the 3 categories (15cm, 130cm, 260cm), and 1,209 live-human samples. We did not experiment with distances that are too far from target devices since attackers would have to use very loud volumes, which would be easily noticed.

Table 6: Effects of variances on detection accuracy.

Diversity	Dimension	Test samples	RoC	Acc.(%)	Prec.(%)	Rec.(%)	F1(%)	EER(%)
Distance	15cm	1,920	0.99	99.6	98.51	99.16	98.93	0.72
	130cm	1,919	0.99	99.7	98.18	99.58	98.87	0.85
	260cm	1,920	0.99	99.9	98.01	100	98.99	0.15
Gender	Male	1,940	0.99	98.9	98.07	99.24	98.66	0.69
	Female	2,062	0.99	98.9	97.76	99.49	98.62	0.97

Evaluation results are presented in Table 6. We show that all F1 scores are greater than 98%, and all EERs are less than 1%. For 15cm, Void achieved 99.6% attack detection rate and an EER of 0.72%. For 130cm, Void achieved 99.7% attack detection rate and an EER of 0.85%. For 260cm, Void achieved 99.9% attack detection rate and an EER of 0.15%. Those results demonstrate that distance variations have minimal impact on the performance of Void.

7.6.2 Gender

Since female voices have typically higher fundamental frequencies than male voices [21, 22], the power distribution patterns may also vary between males and females. To analyze the effects of changing gender, we tested Void separately on (1) 1,940 male live-human voice and attack samples, and (2) 2,062 female live-human voice and attack samples. We selected attack samples that were replayed using the V-MODA speaker with 15cm recording distance. Ten fold cross-validation was used to evaluate Void classifiers.

Again, gender variances did not really influence Void's performance (see Table 6): accuracy and F1 scores are greater than 98%, and EER is below 1%.

7.6.3 Loudspeaker types

To demonstrate Void's performance against high quality speakers, we experimented with various types of loudspeakers. For our dataset and the ASVspoof dataset, we used the trained models described in Section 7.6 and 7.1, respectively. For evaluation, we tested those two models separately on the samples collected through each of the speakers listed in Table 7.

Table 7: Void's performance on different loudspeakers.

Dataset	Loudspeaker	Samples	Detection	Acc.(%)
Our dataset	V-MODA	2,198	2,190	99.6
	Logitech	2,002	1,990	99.4
	Yamaha	1,997	1,996	99.9
	Bose	1,997	1971	98.6
	Smart TV	24,282	24,152	99.4
ASVspoof	Dynaudio BM5A	430	399	92.7
	Behringer Truth B2030A studio monitor	1,381	1,313	95.1
	Genelec 6010A studio monitor	198	160	81.1

Void achieved over 98.5% detection accuracy for all the loudspeakers in our dataset. As for the ASVspoof dataset, it showed varying performance against high quality loudspeakers: the detection accuracy for Dynaudio BM5A and Behringer Truth B2030A studio monitor were 92.7% and 95.1%, respectively; the detection accuracy dropped significantly to 81.1% against Genelec 6010A studio monitor.

7.6.4 Cross data training

For cross data training, we trained Void on the live-human voice and replay attack samples collected from one specific dataset, and evaluated the performance of Void against a different *unseen* (with respect to the human participants and

playback device types) dataset. For the training dataset, we used a single, *fixed* set of 26,965 voice samples collected from 20 male participants, and replayed through the V-MODA speakers. For testing, we considered the following four scenarios: (1) we used 20 unseen male participants' voice samples to perform replay attacks; the V-MODA speaker was used as a playback device; (2) we used voice samples collected from 20 unseen female participants, and replayed them through the V-MODA speaker; (3) we used voice samples collected from 20 unseen female participants, and replayed them through the Bose and Yamaha unseen speakers; and (4) we used voice samples collected from 20 unseen male participants, and replayed through the Bose, Yamaha, and Logitech unseen speakers.

Table 8: Effects of cross training on detection accuracy.

Diversity	Dimension	Test samples	RoC	Acc.(%)	EER(%)
Cross data	Scenario 1	29,956	0.99	100	0.04
	Scenario 2	28,224	0.98	96.4	1.9
	Scenario 3	58,062	0.98	82.1	4.8
	Scenario 4	58,956	0.99	93.2	3.1

We only changed one variable in the first two scenarios but changed all variables in the third and fourth scenario. Table 8 shows the evaluation results for those scenarios. For scenario 1, Void achieved 100% attack detection rate and an EER of 0.04%. For scenario 2, Void achieved 96.4% attack detection rate and an EER of 1.9%. For scenario 3, Void achieved 82.1% attack detection rate and an EER of 4.8%. For scenario 4, Void achieved 93.2% attack detection rate and an EER of 3.1%. As demonstrated from the detection accuracy reductions in scenarios 3 and 4, the performance of Void would degrade as we introduce more variances.

7.7 Replay attacks in unseen conditions

To test Void under various *unseen* and *unexpected* environmental conditions, we installed the speakers and recording devices in an office building. This common area consists of meeting rooms, elevators, entrances and exits, rest rooms, dining areas, information desks, and so on. We replayed all human voice samples (see Section 6.1) on 5 different playback speakers: Galaxy S8 and S9, V-MODA, Bose, and Logitech speakers. We replayed the voice samples using two different volumes, normal and loud, and recorded them using two Galaxy S8 phones, located 30cm and 140cm away from the speakers. The entire recording sessions took about 10 full days to complete. After removing improperly recorded samples, we were left with 119,996 replay attack samples with a huge variety of background noises and situations.

We evaluated the performance of Void against those unseen replay attack samples. Even with such an unexpected and diverse set of replay configurations, Void was able to correctly detect 96.2% of the attacks, showing its robustness in unseen conditions.

8 Robustness against adversarial attacks

We evaluated Void against hidden/inaudible voice command, voice synthesis, EQ manipulation attacks, and combining replay attacks with live-human voices. To measure the attack detection rates, we trained a Void classifier with all of our own replay attack and human voice datasets (see Section 6), and used that classifier to classify given set of attack samples described below. The detection rates of Void against all adversarial attacks are presented in Table 9.

Table 9: Detection rates against adversarial attacks.

Attack	Dataset	# Samples	Acc. (%)
Hidden	Our dataset	1,250	99.7
Inaudible	Ultrasonic speaker	311	100
Synthesis	Our Tacotron dataset	15,446	90.2
EQ manipulation	Strategy 1	350	89.1
	Strategy 2	430	86.3
Combining	Our dataset with human noise	3,600	98.2

8.1 Hidden voice command attacks

Hidden voice commands refer to commands that can not be interpreted by human ears but can be interpreted and processed by voice assistant services [24, 25]. Hidden voice commands typically add more noise-like frequencies to original voice samples during obfuscation, which should increase the overall signal power linearity.

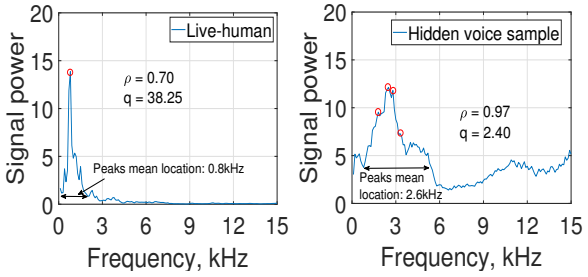

Figure 9: Power spectrum and spectral features representing live-human voice (left) and hidden voice (right) for a sample utterance "*Artificial intelligence is for real.*"

Figure 9 compares the signal power distributions for live-human voice and hidden voice command generated with a phrase *"Artificial intelligence is for real."* The original command is shown on the left, and the obfuscated hidden command, which was played through a loudspeaker, is shown on the right. Unlike the live-human case in which the power distribution shows a non-linear behavior (mostly concentrated below 2 kHz), the linearity coefficients for the hidden voice

samples indicate a more linear behavior (i.e., ρ: 0.97 and q: 2.40). The high power frequency characteristics are also different, which is another indicator for a replay attack.

To evaluate Void against hidden command attacks, we recorded hidden voice command samples using the black-box attack methods demonstrated in [25]. We used 1,250 samples from our own replay attack dataset to generate the attack samples. Void was highly effective against hidden voice command attacks, demonstrating attack detection rate of 99.7% for our replay attack dataset (see Table 9).

8.2 Inaudible voice command attacks

Inaudible voice command attacks involve playing an ultrasound signal with spectrum above 20kHz, which would be inaudible to human ears. Inaudible voice commands are typically played through ultrasonic speakers. Due to the non-linear characteristics of hardware – microphones in this case – the received voice signals are shifted to lower frequencies (down-modulation) with much lower power. To evaluate the performance of Void against inaudible attacks, we implemented an inaudible attack with 347 popularly used Amazon Alexa commands, targeting Echo Dot as the consumer device. We used Google's Text to Speech service (https://pypi.org/project/gTTS/) to convert text commands into speech data. We then modulated voice commands using amplitude modulation with high level frequency of 21kHz. After modulation, the "BatSound L400 ultrasound speaker" (http://batsound.com/?p=12) was used to replay the modulated voice samples. 311 out of 347 commands were successfully recognized and processed by Amazon Alexa. We stored those 311 samples in the ".M4A" file format and used them as the attack set. Void achieved 100% detection rate against inaudible voice command attacks (see Table 9).

8.3 Voice synthesis attacks

To test Void's performance against voice synthesis attack, we used open source voice modeling tools called "Tacotron" [1] and "Deepvoice 2" [2] to train a user voice model with 13,100 publicly available voice samples (https://keithito.com/LJ-Speech-Dataset/). We then used the trained model to generate 1,300 synthesis voice attack samples by feeding in Bixby commands as text inputs.

After attack data generation, we played those synthesis attack samples through four different speakers: Galaxy S8, V-MODA, Logitech 2.1 Ch., and Yamaha 5.1 Ch. speakers were used. For each speaker type, we placed Galaxy S8 in three different distances as described in Section 6.2, and recorded synthesis attack samples. After removing samples that were not properly recorded, we were left with a final set of 15,446 synthesis attack samples and tested them on Void.

Void achieved 90.2% attack detection rate against this set, demonstrating its potential in detecting voice synthesis attacks. However, we note that this is a preliminary result, and further tests need to be performed with test sets generated through models trained on more users.

8.4 Audio EQ manipulation attacks

Since Void leverages spectral power patterns for attack detection, an advanced attacker who is aware of the classification features used by Void may try to craft attack commands using audio EQ programs. EQ manipulation is a process commonly used for altering the frequency response of an audio system by leveraging linear filters. An attacker's goal would be to artificially create attack commands that show power patterns similar to those of live-human voices. By leveraging audio equalization, an attacker could intentionally manipulate the power of certain frequencies to mimic spectrum patterns observed in live-human voices.

To demonstrate the robustness of Void against such EQ manipulation attacks, we used Audacity (https://www.audacityteam.org/) to generate audio samples that mimic decay and peak patterns in spectral power like live human voices under the following two strategies.

The first attack strategy involved removing background noises from audio samples because the samples were originally recorded with various background noises present (e.g., noises generated from fans, refrigerators, or computers). To reduce noise in samples, we used noise reduction rate of 12 dB, and set frequency smoothing parameter to 3. We then boosted power in frequencies less than or equal to 500Hz, and reduced power in frequencies above 500Hz to mimic the characteristics of live-human voices. Using 350 attack samples from the ASVspoof dataset, we *manually* crafted 350 EQ manipulation attack samples based on this power manipulation technique. Void correctly classified 89.1% of them as attacks. The second attack strategy involved applying *bass boost* to increase power in low frequencies between 20Hz and 100Hz to about an average power of 9.5 dB. This power increase would produce more fluctuations in the low frequencies and power patterns similar to those of live-human voices. Audio signals are then normalized with maximum amplitude. Finally, a low pass filter (frequency 1kHz) is applied. We used 430 attack samples from the ASVspoof dataset, and manually crafted 430 EQ manipulation attack samples using this technique. Void correctly classified 86.3% of them as attacks.

We found that the performance of Void was rather degraded against EQ manipulation attacks. However, based on our manual EQ manipulations, we realized that it is quite hard to intentionally craft power patterns that mimic the patterns of live-human voices because most loudspeakers add their own non-linear distortions at low frequencies that cannot easily be controlled by attackers [34]. For instance, it is difficult to craft a sound signal that has desired power peaks at certain fre-

quency ranges even with dedicated manipulation of spectral power patterns.

8.5 Combining replay attacks with live-human voices

To evade detection by Void, attacker can try to simply combine replay attacks with live human voices. For instance, when a command is playbacked through a loudspeaker, a live-human can start uttering some random phrases/commands at the same time.

To analyze the effects of adding live-human voices while replaying attack commands (i.e., both replayed commands and human voices are simultaneously received by a target voice assistant), we recorded additional replay attack samples with two people – both males – continuously chatting near the recording devices. We randomly selected 20 voice samples recorded from 6 participants, and used 6 playback speakers to perform replay attacks: Galaxy S8/S9, Bose, V-MODA, Logitech, and Yamaha speakers. We used three Galaxy S8 and three S9 recording devices, which were spread out and located 1-2m away from the loudspeakers. The two people were sitting about 1-2m away from the recording devices, continuously chatting with their normal voices throughout all recording sessions. Since Void is not responsible for classifying commands that are not properly recognized by voice assistants, we ran all recorded samples through a speech to text translation engine ("Google Speech Recognition"), and removed commands that it failed to recognize – we were left with 3,600 attack samples to test with.

Among those samples, Void correctly detected 3,536 attack samples, achieving a detection accuracy of 98.2%. This result shows that overlapping live-human utterances cannot significantly affect the detection accuracy.

9 Discussion

9.1 Latency and accuracy requirements

The ASVspoof 2017 competition did not measure model and feature complexity nor time taken to train and classify given voice samples – the primary objective was to achieve lowest possible EERs. Consequently, most of the submitted solutions [7] used multiple deep learning models (as an ensemble solution) and heavy classification features to minimize the EERs – such solutions sit uneasily with real-world near-zero-second latency and model complexity requirements.

As shown from our latency results (see Section 7.4), Void is much lighter, faster, and simpler than other top performing solutions as well as the baseline CQCC-GMM solution – many ensemble solutions used CQCC-GMM as the baseline model. Void uses a single classification model and just 97 features. Compared to STFT-LCNN, Void uses 153 times less memory and is about 8 times faster in detection. Void is 1.8 times faster and uses 87 times less memory compared to the baseline CQCC-GMM solution. Void's feature size and testing time performances (shown in Section 7.4) sit more comfortably with the near-zero-second latency and model complexity requirements. While being lightweight, Void still achieves an EER of 11.6%, ahead of the second best solution in the ASVspoof competition [10]. Although this is higher than the 10% EER requirement, our ensemble solution that uses MFCC-GMM (MFCC is moderately light, and is already being used by speech recognition services) achieves 8.7% EER, and satisfy the EER requirement. Further, we demonstrated 0.3% EER against our own dataset.

Being mindful of how light Void is, another possible deployment scenario would involve deploying the Void classifier at the device level: when a user submits a voice command, the voice assistant running on the user's device would first make a voice liveness decision, and drop attack-like commands immediately. With this type of on-device deployment, we would not introduce any new detection (computational) burden on servers.

9.2 Low-incidence population

In practice, Void would be used by a low-incidence population where the number of replay attacks being performed would be much smaller than the number of legitimate uses. Even if Void is configured with a threshold value to minimize false rejection rates (e.g., lower than 5%), users might be annoyed by infrequent voice command rejections. Hence, when an attack is detected, users should be asked to perform explicit authentication (e.g., uttering voice passwords) to still proceed with a suspicious command if authentication is successful. Further, Kwak et al. [33] shows that about 90% of existing mobile voice assistant users use less than 20 commands per month – for those light users, there will only be about five falsely rejected commands every 5 months of use.

However, the incidence level would be quite different when voice assistants are used in homes, e.g., through a smart speaker. This is because there would be frequent loudspeaker noises being generated from TV speakers (e.g., [13]) and standalone speakers. Voice assistants would be stressed with much larger volumes of loudspeaker noises (e.g., TV programs or music being played), and be expected to accurately detect and disregard them. Accurate detection and filtering of loudspeaker noises would improve the reliability of using voice assistants at homes (lower false acceptances), and significantly improve efficiency of speech to text translation engines as loudspeaker noises would not be analyzed.

9.3 Limitations

We tested Void against the ASVspoof dataset (see Section 6.3), which consists of 26 different playback devices and 25 different recording devices (microphones), including studio moni-

tors and headsets, and studio-quality condenser microphones. Our results in Section 7.6.3 show that Void's performance could be downgraded when high quality speakers, such as expensive studio monitors, are used to replay recorded samples. Our audio EQ manipulation attack results (see Section 8.4) showed that carefully crafted adversarial attacks that involve altering frequency responses, or exploiting SVM gradients may be performed to compromise Void. However, such attacks would require strong signal processing expertise.

10 Related work

Recent studies have demonstrated that voice assistants are prone to various forms of voice presentation attacks [6, 11, 12, 20, 23, 33]. Carlini et al. [24, 25] presented hidden voice command attacks to generate *mangled* voice commands that are unintelligible to human listeners but can still be interpreted as commands by devices. Zhang et al. [18] extended this attack to make voice commands completely inaudible by modulating voice commands on ultrasonic carriers. To overcome the limitations of short attack ranges of inaudible attacks (works within about *5ft*) [18, 19], Roy et al. [20] demonstrated the feasibility of launching such attacks from longer distances (i.e., within *25ft* range) by using multiple ultrasonic speakers. They striped segments of voice signals across multiple speakers placed in separate locations.

Many approaches have been proposed to detect machine-generated voice attacks. "VoiceLive" [27] measures the "time difference of arrival" changes in sequences of phoneme sounds using dual microphones available on smartphones. The measured changes are used to determine the sound origin of each phoneme (within the human vocal system) for liveness detection. VoiceLive was evaluated with 12 participants, demonstrating 1% EER. Zhang et al. [3] also proposed articulatory gesture-based liveness detection (analyzing precise articulatory movements like lip and tongue movements); their approaches, however, are only applicable to scenarios where a user is physically speaking near a smartphone's microphone. In contrast, Void would work well even when users are a few meters (speaking distances) away from target devices. Chen et al. [12] leveraged magnetic fields emitted from loudspeakers to detect replay attacks. Their approach, however, requires users to utter a passphrase while moving smartphones through a predefined trajectory around sound sources. Blue et al. [14] found that the amount of energy in a sub-bass region (between 20Hz and 250Hz) can be used to distinguish live human voices from speaker generated voices. However, their approach relies on being aware of ambient noise power as a priori while performing noise filtering – this is necessary to measure the amount of energy with high accuracy. Therefore, this technique could be compromised by intentionally controlling the ambient noise power that the noise filtering is relying on. Feng et al. [11] proposed a voice authentication system that uses a wearable device, such as eyeglasses – collecting a user's body surface vibrations, and matching it with voice signals received by a voice assistant through a microphone. Although their approach is capable of achieving about 97% accuracy, they rely on an additional hardware that users have to carry.

An extensive study was conducted to analyze the performances of machine learning-based replay attack detection techniques proposed as part of the 2017 ASVspoof competition [7]. According to the study findings, the equal error rates (EER) varied from 6.7% to 45.6% [30] – most solutions used an ensemble approach, and used CQCC-GMM as a baseline model, which alone is complex and uses about 14,000 features. We implemented STFT-LCNN [30], which is one of the two deep learning models used by the top performing solution from the competition. Our evaluations showed that despite its low EER, STFT-LCNN alone is unacceptably heavy and slow. Likewise, existing solutions have been designed merely to minimize EERs.

Tom et al. [17] achieved 0% EER on the ASVspoof evaluation set using Residual Network as the deep learning model and group delay grams as the classification features. Group delay (GD) grams are obtained by adding a group delay function over consecutive frames as a time-frequency representation. However, they used another external dataset to pre-train a model, and applied transfer learning technique on that model using the ASVspoof train set. Since their model training methods and assumptions are not consistent with how ASVspoof models are suppose to be trained (i.e., they assume other datasets are available), we do not directly compare Void with their solution. We implemented their model as close as possible to the descriptions provided in the paper, and analyzed its time and space complexity as described in Appendix I. Their model uses 786,432 features compared to Void's 97 features, and uses about 1,195MB of memory on average for classifying a sample. Void uses just 2MB.

Consequently, multiple complex models and heavy features have been used without considering any latency and model complexity requirements described in Section 3.1. Void was designed to use a small number of features, and guarantee fast training and classification times as well as model simplicity. Further, all the literature discussed above present model structure and accuracy results without providing insights into the spectral power features and their characteristics – replying on deep learning techniques for feature extraction has this limitation. In contrast, we explain the spectral power patterns and non-linearity for loudspeaker detection as part of feature engineering (see Section 4).

11 Conclusion

Void analyzes the spectral power patterns of voice signals to accurately detect voice replay attacks. Compared with existing methods using multiple, heavy classification models, Void runs on a single efficient classification model with 97 features

only, and does not require any additional hardware.

Our experiments, conducted on two large datasets collected under numerous varying conditions (demographics, speaker/microphone types, and background noises), showed that Void can achieve 0.3% EER on our own dataset, and 11.6% EER on the ASVspoof evaluation set. On average, Void took 0.03 seconds to classify a given voice sample, and used just 1.98 megabytes of memory. Void is about 8 times faster, and 153 times lighter (with respect to feature size) than the top performing LCNN-based solution. Also, our ensemble solution (that uses moderately light, already available MFCC features) achieves 8.7% EER – making it a much more practical, and attractive solution for businesses to consider. Moreover, Void is resilient to adversarial attacks including hidden command [24, 25], inaudible voice command [18–20], voice synthesis [6, 12], EQ manipulation, and combining replay attacks with live-human voices achieving over 86% detection rates for all of those attack types.

Acknowledgment

This work was supported by Samsung Research and NRFK (2019R1C1C1007118). The authors would like to thank all the anonymous reviewers and Carrie Gates for their valuable feedback. Note that Hyoungshick Kim is the corresponding author.

References

[1] Y. Wang, R.J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, R. A. Saurous, "Tacotron: Towards End-to-End Speech Synthesis", in *Proceedings of the 18th INTERSPEECH*, 2017.

[2] A. Gibiansky, S. Arik, G. Diamos, J. Miller, K. Peng, W. Ping, J. Raiman, Y. Zhou, "Deep Voice 2: Multi-Speaker Neural Text-to-Speech", in *Advances in Neural Information Processing Systems 30*, pp. 2966-2974, 2017.

[3] L. Zhang, S. Tan, J. Yang, "Hearing Your Voice is Not Enough: An Articulatory Gesture Based Liveness Detection for Voice Authentication", in *Proceedings of the 24th ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[4] S. McCandless, "An algorithm for automatic formant extraction using linear prediction spectra", in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 22, no. 2, pp. 135-141, 1974.

[5] T. F. Li, S. Chang, "Speech recognition of mandarin syllables using both linear predict oding cepstra and Mel frequency cepstra", in *Proceedings of the 19th Conference on Computational Linguistics and Speech Processing*, 2007.

[6] A. Janiki, F. Alegre, and N. Evans, "An assessment of automatic speaker verification vulnerabilities to replay spoofing attacks", in *Security and Communication Networks*, pp. 3030-3044, 2016.

[7] T. Kinnunen, M. Sahidullah, H. Delgado, M. Todisco, N. Evans, J. Yamagishi, and K. A. Lee, "The ASVspoof 2017 Challenge: Assessing the Limits of Replay Spoofing Attack Detection", in *Proceedings of the 18th INTERSPEECH*, 2017.

[8] H. Delgado, M. Todisco, M. Sahidullah, N. Evans, T. Kinnunen, K. A. Lee, J. Yamagishi, "ASVspoof 2017 Version 2.0: meta-data analysis and baseline enhancements", in *Proceedings of the Speaker and Language Recognition Workshop*, 2018.

[9] T. Kinnunen, N. Evans, J. Yamagishi, K. A. Lee, M. Sahidullah, M. Todisco, and H. Delgado, "ASVspoof 2017: Automatic Speaker Verification Spoofing and Countermeasures Challenge Evaluation Plan", [Online:] https://www.asvspoof.org/data2017/asvspoof-2017_evalplan_v1.2.pdf, 2017.

[10] T. Kinnunen, M. Sahidullah, H. Delgado, M. Todisco, N. Evans, J. Yamagishi, and K. A. Lee, "The ASVspoof 2017 Challenge: Assessing the Limits of Replay Spoofing Attack Detection", [Online:] https://www.asvspoof.org/slides_ASVspoof2017_Interspeech.pdf, 2017.

[11] H. Feng, K. Fawaz, and K. G. Shin, "Continuous Authentication for Voice Assistants", in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, 2017.

[12] S. Chenyz, K. Reny, S. Piaoy, C. Wang, Q. Wangx, J. Weng, L. Suy, and A. Mohaisen, "You Can Hear But You Cannot Steal: Defending against Voice Impersonation Attacks on Smartphones", in *Proceedings of IEEE 37th International Conference on Distributed Computing Systems*, 2017.

[13] A. Liptak, "Amazon's Alexa started ordering people dollhouses after hearing its name on TV", [Online:] https://www.theverge.com/2017/1/7/14200210/amazon-alexa-tech-news-anchor-order-dollhouse, 2017.

[14] L. Blue, L. Vargas, and P. Traynor, "Hello, Is It Me You're Looking For?: Differentiating Between Human and Electronic Speakers for Voice Interface Security" in *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2018.

[15] D. Luo, H. Wu, and J. Huang, "Audio recapture detection using deep learning", in *Proceedings of the 3rd IEEE China Summit and International Conference on Signal and Information Processing*, 2015.

[16] Sound Engineering Academy, "Human Voice Frequency Range", [Online:] http://www.seaindia.in/blog/human-voice-frequency-range/

[17] F. Tom, M. Jain and P. Dey, "End-To-End Audio Replay Attack Detection Using Deep Convolutional Networks with Attention", in *Proceedings of the 19th INTERSPEECH*, 2018.

[18] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, W. Xu, "DolphinAtack: Inaudible Voice Commands", in *Proceedings of the 24th ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[19] L. Song, P. Mittal, "POSTER: Inaudible Voice Commands", in *Proceedings of the 24th ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[20] N. Roy, S. Shen, H. Hassanieh, R. R. Choudhury, "Inaudible Voice Commands: The Long-Range Attack and Defense", in

Proceedings of 15th USENIX Symposium on Networked Systems Design and Implementation, 2018.

[21] I. R. Titze, "Principles of Voice Production", in *Prentice Hall*, 1994.

[22] R. J. Baken, "Clinical Measurement of Speech and Voice", in *Taylor & Francis*, 2000.

[23] S. Panjwani and A. Prakash, "Crowdsourcing Attacks on Biometric Systems", in *Proceedings of the 10th Symposium On Usable Privacy and Security*, 2014.

[24] T. Vaidya, Y. Zhang, M. Sherr, and C. Shields, "Cocaine Noodles: Exploiting the Gap between Human and Machine Speech Recognition", in *Proceedings of the 9th USENIX Workshop on Offensive Technologies*, 2015.

[25] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden Voice Command", in *Proceedings of the 25th USENIX Security Symposium*, 2016.

[26] P. Castiglioni, "Levinson-durbin algorithm", in *Encyclopedia of Biostatistics*, 2005.

[27] L. Zhang, S. Tan, J. Yang, and Y. Chen, "VoiceLive: A Phoneme Localization based Liveness Detection for Voice Authentication on Smartphones", in *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[28] L. Breiman, "Random Forests", in *Machine Learning*, vol. 45, no. 28, pp. 5–32, 2001.

[29] ASVspoof, [Online:] https://datashare.is.ed.ac.uk/handle/10283/2778

[30] G. Lavrentyeva, S. Novoselov, E. Malykh, A. Kozlov, O. Kudashev, V. Shchemelinin, "Audio replay attack detection with deep learning frameworks", in *Proceedings of the 18th INTERSPEECH*, 2017.

[31] X. Zhao, Y. Wang, and D. Wang, "Robust speaker identification in noisy and reverberant conditions", in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[32] G. Valenti, A. Daniel, and N. Evans, "End-to-end automatic speaker verification with evolving recurrent neural networks", in *Odyssey 2018 The Speaker and Language Recognition Workshop*, 2018.

[33] I. Kwak, J. Huh, S. Han, I. Kim, and J. Yoon, "Voice presentation attack detection through text-converted voice command analysis", to appear in *ACM CHI Conference on Human Factors in Computing Systems*, 2019.

[34] W. Frank and R. Reger and U. Appel, "Loudspeaker nonlinearities-analysis and compensation", in *Conference Record of the Twenty-Sixth Asilomar Conference on Signals, Systems Computers*, 1992.

[35] The Audibility of Distortion At Bass Frequencies, [Online:] https://www.audioholics.com/loudspeaker-design/audibility-of-distortion-at-bass, 2015.

[36] P. Gil-Cacho, T. V. Waterschoot, and M. Moonen, and S. Jensen, "Study and characterization of odd and even nonlinearities in electrodynamic loudspeakers", in *Proceedings of the 127th Audio Engineering Society*, 2009.

[37] V. Gunnarsson, "Assessment of nonlinearities in loudspeakers", in *Chalmers University of Technology*, 2010.

A Classifying live-human voices and voices replayed through in-built speakers with three signal power features

Figure 10 shows the spectral power features (power-sum in each frequency) of 800 voice samples: 400 were live-human samples, and the other 400 were samples replayed through 11 in-built smartphone speakers. As shown in Figure 10, three signal power features, μ_{peak}, ρ and q, look noticeably different, suggesting that they could be effective in classifying live-human voices and in-built speakers (those features are explained in Section 5.3).

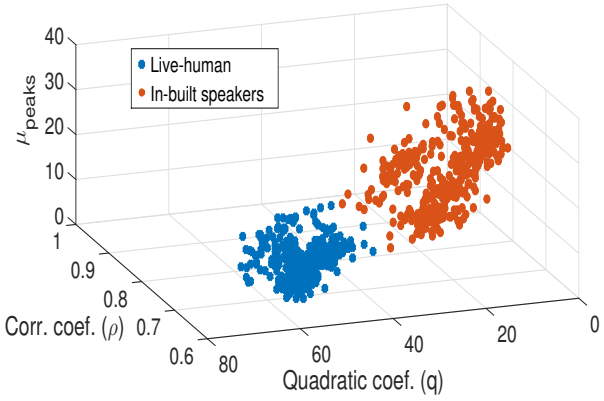

Figure 10: Integral signal power features used to classify live-human voices and voices replayed through 11 in-built smartphone speakers.

B Power patterns for different loudspeakers

Figure 11 shows power patterns for a live-human voice and 8 different loudspeakers (from our dataset and ASVspoof 2017 dataset). In the live-human voice sample (top left), there are four distinct peaks in the power pattern below 2 kHz. Except for `Genelec 6010A studio monitor`, and `Focus Scarlett 2i2 audio`, all other high quality speakers show a single sharp peak or small peaks only in their power patterns. As for Genelec and Focus Scarlett speakers, the power patterns below 2 kHz are similar to those of live-human patterns. To deal with such studio-level quality speakers, Void employs other feature sets as explained in Section 5.3.

C Summary of linearity degree features

For the linearity degree of pow_{cdf}, we compute the following two features: Pearson correlation coefficients ρ and quadratic curve fitting coefficients q of pow_{cdf} (see Table 10).

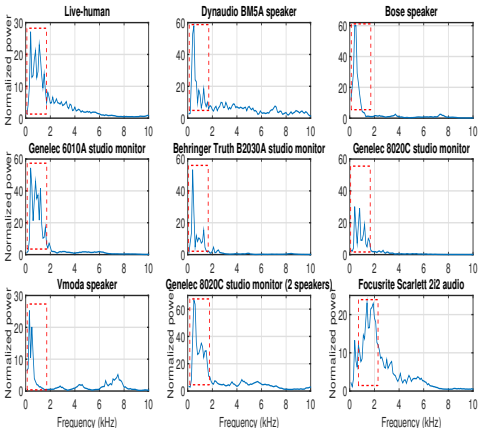

Figure 11: Power patterns of live-human and different loudspeakers.

Table 10: Summary of the linearity degree features.

Features	Symbol
Cross-correlation coefficients	ρ
Quadratic curve-fitting coefficients	q
FV_{LDF}= {ρ, q}	

We use Pearson correlation coefficient ρ to measure of the linearity in the signal power pattern. The Pearson correlation coefficients can be calculated as:

$$\rho(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y}, \quad (1)$$

where cov is the covariance, and σ_X and σ_Y represent the standard deviations of X and Y, respectively. In our experiments $X = pow_{\text{cdf}}$ and Y is an increasing sequence $\{y_n\}$, where $y_{n+1} - y_n = 1$.

A polynomial $q(x)$ of degree $n = 2$ with respective coefficients are given below as:

$$q(x) = q_1 x^2 + q_2 x + q_3, \quad (2)$$

where $x = pow_{\text{cdf}}$ in the above equation. We use the quadratic coefficient q_1 in our features which is denoted by q for simplicity.

We measure the signal power linearity to show the difference in power patterns between live-human and in-built loudspeakers. Table 11 shows mean and standard deviation of the linearity features of 400 live-human samples and 400 samples replayed through in-built speakers, respectively.

D Summary of high power frequency features

Given the vector $< pow >$ of power density values and the peak selection threshold ω, we compute the feature vector (FV_{HPF}) to capture the dynamic characteristics of spectral power in higher frequencies (see Table 12).

Table 11: Means and standard deviations of signal power linearity features for live-human and in-built speakers.

Source	Feature	mean	stdev
Live-human	ρ	0.759	0.059
	q	47.960	6.541
In-built speakers	ρ	0.854	0.053
	q	10.267	7.006

Table 12: Summary of the high power frequency features.

Features	Symbol
#peaks in high-power frequencies	N_{peaks}
Relative frequencies corresponding to *peaks*	μ_{peaks}
Standard deviation of high power frequency location	σ_{peaks}
$FV_{HPF} = \{N_{peaks}, \mu_{peaks}, \sigma_{peaks}\}$	

Table 13 shows the analysis of those three key features for 6,362 voice samples replayed through 13 standalone speakers, and 3,558 live-human voice samples. The mean number of peaks (N_{peaks}) for live-human voices is significantly greater than those of standalone speakers. Similarly, live-human voices showed greater mean of relative frequencies corresponding to *peaks* (μ_{peaks}) and standard deviations. These difference could be analyzed to detect standalone speakers.

Table 13: Means and standard deviations of the high power frequency features for live-human and standalone speakers.

Source	Features	mean	stdev
Live-human	N_{peaks}	2.580	3.029
	μ_{peaks}	7.377	2.693
Standalone speakers	N_{peaks}	1.695	1.348
	μ_{peaks}	5.531	2.110

E Finding the optimal feature set

Table 14 shows a separate evaluation result for different feature sets. We used the ASVspoof 2017 dataset for evaluation. We used the train and development sets for training, and used the evaluatoin set for testing. The results show that each of the selected feature set achieves an F1-score greater than 80%. These results, together with the declining EERs observed with addition of features, demonstrate that all individual features (FV_{LFP}, FV_{LDF}, FV_{HPF}, and FV_{LPC}) are integral in achieving an EER of 11.60%.

Table 14: Accuracy evaluation for each selected feature set (see Section 5.3)

Feature set	Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)	EER (%)
FV_{LFP}	76.61	75.59	98.04	85.37	19.37
FV_{LDF}	72.14	72.91	95.06	82.52	30.92
FV_{HPF}	73.13	71.61	98.09	82.79	21.47
FV_{LPC}	70.19	68.62	97.64	80.60	22.99
FV_{LFP} + FV_{LDF} + FV_{HPF}	79.51	79.08	97.79	87.44	18.83
FV_{LFP} + FV_{LDF} + FV_{HPF} + FV_{LPC} (Void)	84.33	83.51	98.96	90.58	11.60

F Feature and model parameters

We describe parameters used for recording voice samples, performing feature engineering, and training classifiers. We used sampling frequency of 44.1kHz for voice recording. As for the STFT parameters, we used 1024 as the window length (recommended to be power of 2), 256 as the hop size, and 4,096 as the number of FFT points. Other parameters needed to train Void are presented in Table 15.

Table 15: Feature and model parameters.

Class	Parameter	Value
Voice	Sampling frequency	44.1kHz
	Window length	1024
	Hop length	256
	nfft	496
Void	W	10
	ω	0.6
	pow_{cdf}'s polynomial order for estimating q	2
	P_{est}'s estimation using polynomial order	6
SVM	Kernel	RBF
	Kernel scale	Auto

G List of playback devices

We used 11 different types of in-built speakers including smartphones and a smart TV, and four standalone speakers to replay recorded voice samples (see Table 16).

H List of recording devices

We used 3 different laptops, and 9 different smartphones as recording devices (see Table 17).

I Implementation of GD-ResNet

Based on the model described in [17], we implemented GD-ResNet with two stages: the first stage is used to estimate attention weights from a Global Average Pooling layer, and the second stage is used to train a ResNet-18 model based on the GD gram feature with attention weights. Table 18 summarizes the performance of our GD-ResNet implementation: it achieved 0% and 23% EERs on our own dataset and the ASVspoof 2017 dataset, respectively. As for space complexity, we counted the number of features extracted from a single voice sample. Compared to 97 features used by Void, GD-ResNet uses 786,432 features. As for the average memory used for classifying a sample, Void uses about 1.99MB, whereas GD-ResNet uses 1,194.68MB.

Table 16: List of playback devices (loudspeakers) used for replay attack dataset generation.

	Name	Model
In-built	Galaxy A8	A810S
	Galaxy A5	SM-A500x
	Galaxy Note 8	SM-N950x
	Galaxy S8	SM-G950
	Galaxy S8	SM-G955N
	Galaxy S9	SM-G960N
	iPhone SE	A1662
	iPhone 6S Plus	A1524
	iPhone 5S	A1519
	LG V20	V20 F800
	Samsung Smart TV	QN49Q7FAM
Standalone	Bose	SoundTouch 10
	V-MODA	REMIX-BLACK
	Logitech (2.1 Ch.)	Z623
	Yamaha (5.1 Ch.)	YHT-3920UBL

Table 17: List of recording devices used for human voice collection, and replay attack dataset generation.

Maker	Model
Samsung Notebook	NT910S3T-K81S
Samsung Notebook	NT200B5C
Macbook Pro	A1706 (EMC 3163)
Galaxy A5	SM-A500x
Galaxy Note 8	SM-N950x
Galaxy S8	SM-G950
Galaxy S8	SM-G955N
Galaxy S9	SM-G960N
iPhone SE	A1662
iPhone 5S	A1519
iPhone 6S Plus	A1524
LG V20	V20 F800

Table 18: GD-ResNet space complexity.

Measure	Void	GD-ResNet[17]
Extraction (sec.)	0.035	0.100
Training (sec.)	0.283	40,560.264
Testing (sec.)	0.035	0.120
#Features	97	786,432
Memory size (MB)	1.988	1,194.684
Performance (EER)	11.6%	23%