DEFENDING AGAINST ADVERSARIAL ATTACKS IN SPEAKER VERIFICATION SYSTEMS

by

Li-Chi Chang

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science

Department of Computer Science

Fort Wayne, Indiana

August 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Zesheng Chen, Chair

Department of Computer Science

Dr. Jin Soung Yoo

Department of Computer Science

Dr. Bin Chen

Department of Electrical and Computer Engineering

Approved by:

Dr. Jin Soung Yoo

To God Almighty, my strength and my shield.

To the helpers from Him.

To my wife Pei-Yu Tang.

ACKNOWLEDGMENTS

I am very grateful to Dr. Zesheng Chen, who is my advisor and gave me a lot of guidance, encouragement, and various supports in every aspect of this project. He shared his knowledge and comprehensive comments with me and helped me formulate my thinking. He discussed with me to apply intuition and reasoning, gave me critical feedbacks, and emphasized the quality of the project. Moreover, he gave many suggestions to my first thesis in my lifetime. His support and guidance is far beyond his responsibility. It is my great honor and appreciation to work with Dr. Zesheng Chen during my master study.

I want to thank Dr. Jin Soung Yoo and Dr. Bin Chen for being my master thesis committee members. Their comments and feedbacks help improve the project. I also thank the support from PFW Graduate Research Assistantship and PFW Collaborative Research Grant. Moreover, I would like to thank Dr. Guoping Wang for allowing me to access the GPU server to perform the experiments.

Finally, I appreciate the support and the care from my family and my spiritual family in Christ to help me overcome my stress.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

With the advance of the technologies of Internet of things, smart devices or virtual personal assistants at home, such as Google Assistant, Apple Siri, and Amazon Alexa, have been widely used to control and access different objects like door lock, blobs, air conditioner, and even bank accounts, which makes our life convenient. Because of its ease for operations, voice control becomes a main interface between users and these smart devices. To make voice control more secure, speaker verification systems have been researched to apply human voice as biometrics to accurately identify a legitimate user and avoid the illegal access. In recent studies, however, it has been shown that speaker verification systems are vulnerable to different security attacks such as replay, voice cloning, and adversarial attacks. Among all attacks, adversarial attacks are the most dangerous and very challenging to defend. Currently, there is no known method that can effectively defend against such an attack in speaker verification systems.

The goal of this project is to design and implement a defense system that is simple, light-weight, and effectively against adversarial attacks for speaker verification. To achieve this goal, we study the audio samples from adversarial attacks in both the time domain and the Mel spectrogram, and find that the generated adversarial audio is simply a clean illegal audio with small perturbations that are similar to white noises, but well-designed to fool speaker verification. Our intuition is that if these perturbations can be removed or modified, adversarial attacks can potentially loss the attacking ability. Therefore, we propose to add a plugin-function module to preprocess the input audio before it is fed into the verification system. As a first attempt, we study two opposite plugin functions: denoising that attempts to remove or reduce perturbations and noise-adding that adds small Gaussian noises to an input audio. We show through experiments that both methods can significantly degrade the performance of a state-of-the-art adversarial attack. Specifically, it is shown that denoising and noise-adding can reduce the targeted attack success rate of the attack from 100% to only 56% and 5.2%, respectively. Moreover, noise-adding can slow down the attack 25 times in speed and has a minor effect on the normal operations of a speaker verification system. Therefore, we believe that noise-adding can be applied to any speaker verification system against adversarial attacks. To the best of our knowledge, this is the first attempt in applying the noise-adding method to defend against adversarial attacks in speaker verification systems.

1. INTRODUCTION

In recent years, the number of smart devices has been growing exponentially. Different Internet of things (IoT) devices at home, such as smart blobs, smart air conditioner, and smart door lock, can be easily controlled via a smart phone. Moreover, a person can access their bank account through smart devices or virtual personal assistants like Google Assistant [1], Apple Siri [2], Amazon Alexa [3], and Microsoft Cortana [4]. These smart home devices can be accessed and controlled through different methods, such as position detection, habit record, and most importantly, voice control [5]. Because of its convenience and ease for operations, voice control is becoming the main interface at home for controlling smart devices.

To make voice control more secure, speaker verification systems have been widely studied and attempt to apply human voice as biometrics to distinguish people, in a similar way as fingerprint and iris recognition. Comparing with other verification methods, speaker verification has the advantages of being hand free and distance flexible. That is, speaker verification does not require to have the contact with smart devices and is able to operate within a certain distance.

However, speaker verification systems have been facing many different attacks that attempt to compromise the integrity of the systems to allow the illegal access from attackers [6]. The main attacks include replay attacks [7], voice cloning attacks [8], and adversarial attacks [9]. Among all attacks, adversarial attacks are the most dangerous and very difficult to detect and defend. In such an attack, small well-designed perturbations are added to a clean audio from an illegal speaker to form the adversarial audio, which is barely perceptible by humans. That is, a person can hardly distinguish between the original clean audio and the adversarial audio when hearing them. However, the adversarial audio can be falsely accepted by the speaker verification system. As shown in [10], FakeBob adversarial attacks can achieve at least 99% targeted attack success rate on both open source and commercial speaker verification systems. That is, more than 99% of generated adversarial audios can be falsely accepted by the speaker verification systems. On the other hand, many defense systems that perform well against adversarial attacks for the images in the area of the image classification problem, such as local smoothing [11], quantization [11], and temporal dependency detection [11], cannot defeat the FakeBob attacks.

As a result, this is an important research question: How can we effectively and efficiently defend against adversarial attacks such as FakeBob? The goal of this project is to design and

implement a defense system that is simple, light-weight, and effectively against adversarial attacks. Specifically, the defense system should be compatible with an existing speaker verification system and should not require any change to the internal structure of the currently used speaker verification system. Moreover, the proposed system should only slightly increase the computation load of the speaker verification system. Most importantly, the defense system should be able to significantly slow down an attacker to generate a successful adversarial audio or greatly reduce the attack success rate. On the other hand, the defense method should only slightly affect the normal operations of a speaker verification system.

To achieve the goal of the project, we start with studying the adversarial audio in both the time domain and the Mel spectrogram [12]. We find that the perturbations are similar to white noises, but they are not random and are intentionally designed to fool the speaker verification systems. Based on these observations, our intuition is that if the perturbations in an adversarial audio can be removed or modified, adversarial attacks would loss the efficiency against the speaker verification system. That is, before an input audio is provided to a verification system, a plugin function is applied to this input audio to preprocess it, in order to reduce the effect of perturbations from attacks. In this work, we consider two different plugin functions: denoising and noise-adding. The basic idea of the denoising function is to remove or reduce the perturbations or noises in input audios; and the implementation of denoising is based the work from [13]. The goal of the noise-adding is to append some small Gaussian noises to the input audio to perturb adversarial audios, so that adversarial attacks would loss or reduce the ability to mislead a speaker verification system. We find that denoising and noise-adding are indeed opposite operations. Through experiments using the state-of-the-art speaker verification system such as Gaussian Mixture Model, we find that both methods can significantly reduce the attack success rate of FakeBob, especially the noise-adding method.

The most relevant work to our approach was recently presented in [14], when we are working on this thesis. This work also studied how to use small noises to counteract query-based black-box adversarial attacks. However, there are some key differences between their work and our work: (1) The work in [14] focuses on the image classification problem, whereas we study the speaker verification area. Images and audios are different signals and have distinct characteristics. As shown in [10], many defense systems that perform well for images cannot be applied to audios. (2) Due to different fields, image classification and speaker verification apply different machine

learning or deep learning methods. For example, an image classifier uses the classic convolutional neural network (CNN) model, whereas a speaker verification system applies the Gaussian Mixture Model. The same defense mechanism may have different effects on distinct machine learning models. (3) The work in [14] aims at untargeted adversarial attacks, whereas our work focuses on targeted attacks.

We summarize our main discoveries or contributions in the following:

- We find and show that the perturbations in an adversarial audio are very small and are similar to white noises. On the other hand, these perturbations are not random, but are intentionally designed to fool the speaker verification systems.
- We propose a defense framework that is simple, light-weight, and effectively against adversarial attacks in speaker verification systems.
- We find that the denoising function is able to reduce or remove the perturbations, but at the same time introduce the echo effect to the input audio. As shown in our experiments based on FakeBob [10], denoising can reduce the targeted attack success rate from 100% to 56%. A downside of denoising is that in some speaker verification system, it may affect the processing time nonnegligibly.
- We discover that the noise-adding method performs much better than the denoising function. For example, we show that noise-adding can further reduce the targeted attack success rate of FakeBob to 5.2%. Moreover, the speed for FakeBob to generate an adversarial audio is slowed down for 25 times under the impact of this defense. On the other hand, the processing time of the noise-adding function is very small and can be negligible. Furthermore, noise-adding slightly affects the normal operations of a speaker verification system. Therefore, we believe that such a simple solution can be applied to any speaker verification system against adversarial attacks.

The remainder of this thesis is structured as follows. Section 2 surveys the literatures related to speaker verification and attacks against speaker verification systems. Section 3 provides the background of this project, including voice characteristics, the working process of a state-of-the-art speaker verification system, and the detailed information of adversarial attacks (*i.e.*, FakeBob) against speaker verification systems. Section 4 presents our proposed defense framework and discusses two plugin functions, *i.e.*, denoising and noise-adding, and their impacts on a clean audio.

Section 5 evaluates the performance of our proposed defense system on normal operations of speaker verification systems and against FakeBob adversarial attacks. Finally, Section 6 provides the conclusions and discusses the future work.

2. LITERATURE SURVEY

Speaker verification is an active research area. Many implementations in this field are based on machine learning, especially deep learning [6]. In this section, we first review speech models and speaker verification systems, and then survey security attacks against speaker verification systems.

2.1 Speech Models and Speaker Verification Systems

The speech applications can be divided into three main types: text-to-speech, voice cloning, and speaker verification.

2.1.1 Text-to-Speech Models

Text-to-speech applications attempt to translate a text into a speaker voice. In general, the multiple speaker text-to-speech models have higher versatility than single speaker models [15] [16] [17] [18] [19] [20]. Jia et al. introduced the transfer learning from speaker verification to multiple speaker text-to-speech synthesis [19]. Specifically, the authors combined three main technologies or components: the speaker encoder that transfers the input speaker reference waveform to a low dimensional vector called speaker embedding [21], the synthesizer that uses the input text and the speaker embedding to generate a Mel spectrogram [22], and the vocoder that recovers the Mel spectrogram into the waveform in the time domain [23]. One of the main contributions of this paper [19] is that each of three components is trained independently.

2.1.2 Voice Cloning Models

Voice cloning attempts to synthesize a voice with the features of the targeted speaker's voice. Wang et al. built Tacotron, a deep learning model, to clone the voice of a specific speaker [20]. For Tacotron, it usually requires many hours of the audio of the targeted speaker as the training data. Deep voice series [15] [18] and their variants [16] [17] have extended the voice cloning to multiple speakers. For example, Deep Voice 3 in [18] uses the attention [24] to build a sequence-to-sequence model, extracts the key-value pairs from the text embedding input by a convolution

neural network, applies the queries from the Mel spectrogram of the input audio by another convolution neural network, and finally converts the resulting Mel spectrogram back to the waveform in the time domain by using converters such as Griffin-Lim [25], WORLD synthesis [26], or WaveNet [23]. However, Deep Voice 3 can only generate the cloned voices of speakers whose samples are in the dataset. It cannot do the "0-shot" cloning and cannot generate cloned voices for speakers whose samples are not seen yet. Arik et al. presented the neural voice cloning with a few samples based on Deep Voice 3 in [16]. This model decreases the length of required enrollment audios to several seconds and builds a few-shot voice cloning model. Furthermore, Jung and Kim designed the neural voice cloning with a few low-quality samples in [17]. The authors used a generative adversarial network (GAN) to purify the low-quality samples.

Table 2.1 shows the comparisons of different state-of-the-art voice cloning models.

Table 2.1 *Voice Cloning Models*

Model	Feature	MOS	Enroll Size
Voice Cloning with Few Low-Quality Samples [17]	GAN	N/A	11 sec
Voice Cloning with a Few Samples [16]	0-shot voice cloning	2.5	3.5 sec
Deep Voice 3 [18]	Only speakers in Dataset	3.6	2 hours
Tacotron [20]	Single Speaker	3.8	24.6 hours

In the table, the mean opinion score (MOS) is a subjective measure of an audio clip and has been widely used to evaluate the performance of a voice cloning model. A higher value of MOS reflects a better voice cloning model.

2.1.3 Speaker Verification Systems

Speaker verification systems attempt to determine if a given audio clip is from a registered legitimate speaker or from an illegal user [27] [28] [21]. For a speaker verification system, the main performance metric is the equal error rate (EER), which will be discussed in details in Section 3.2. A smaller value of EER reflects a better speaker verification system. The state-of-the-art speaker verification systems include Gaussian Mixture Model (GMM) [29], i-vector [30], d-vector [31], and x-vector [32]. The GMM model trains a Gaussian mixture model as a universal background model [29], whereas the i-vector model combines joint factor analysis and support

vector machines for speaker verification [30] . Both d-vector and x-vector models are based on deep neural networks [31] [32].

Speaker verification systems can be divided into two types of models: text-dependent and text-independent models. Text-dependent models are based on audios that a speaker says pre-defined sentences, whereas text-independent models do not require the speaker to follow specific sentences. Usually a text-dependent model has a smaller EER than a text-independent model [33].

Recently, different technologies have been applied to build a speaker verification system. For example, Torfi et al. introduced a text-independent speaker verification system by using a series of 3D convolutional neural network layers in [27]. The authors calculated the similarity between enrolled audios and an input audio to verify a speaker. Wan et al. from Google proposed a new loss function, called generalized end-to-end (GE2E) loss in [21]. The authors believe that the loss function is more important than a model. They tested their GE2E in [34] with text-dependent and text-independent speaker verification models and showed that GE2E leads to a better EER. Yan et al. applied the idea of field print to identify if a sound is from a real speaker or from an audio played by a device in [28].

2.2 Attacks against Speaker Verification Systems

Speaker verification systems are vulnerable to different security attacks. Attackers attempt to compromise the integrity of a speaker verification system and make it false accept an illegal access or an illegal audio. Basically, there are two types of attacks: white-box and black-box. In white-box attacks, the attacker has the detailed information of the targeted speaker verification system. On the other hand, the attacker who uses the block-box attacks does not have any information about the internal structure of the system. From the perspective of the attackers, the black-box attacks are much more difficult to implement. As shown in FakeBob attacks [10], the black-box attacks are possible and even very effective against the state-of-the-art speaker verification systems.

The main security attacks against speaker verification systems can be categorized into three types: replay, cloning, and adversarial attacks.

2.2.1 Reply Attacks against Speaker Verification Systems

The replay attack is a simple method that attackers record the voice of a targeted speaker and then reply the sound to fool a text-independent speaker verification system, which is a sniffing and spoofing attack [35]. The success of such an attack depends on that the voice of the targeted speaker can be recorded. Moreover, Yan et al. showed that the field print can be applied to counteract this attack [28].

2.2.2 Cloning Attacks against Speaker Verification Systems

Voice cloning attacks attempt to synthesize the targeted speaker's voice to bypass a text-dependent speaker verification system [15] [16] [17] [18] [19] [20]. The success of such an attack requires that the attacker is able to collect a large amount of the targeted speaker's voice data, which is usually very difficult to implement. As shown in the Table 2.1, if MOS needs to be high, hours of a speaker's voices are required. As a result, voice cloning attacks are not widely applied in reality.

2.2.3 Adversarial Attacks against Speaker Verification Systems

Different from other attacks, adversarial attacks do not require to obtain or record the targeted speaker's voice. Such an attack can fiddle an illegal voice to break into a speaker verification system by adding small well-designed noise-like perturbations.

The study of adversarial attacks started from the research of applying deep learning to the image classification problem. In their senior work, Szegedy and Goodfellow et al. showed that deep learning is particularly vulnerable to adversarial examples attacks [9] [36]. For example, after adding very small perturbations, the image of panda can be recognized as gibbon with 99.3% confidence by a popular deep-learning based classifier. Later, researcher realized that adversarial attacks can be applied to not only deep learning, but also other machine learning methods. Please refer to the paper [37] for a comprehensive survey on adversarial attacks.

Recently, adversarial examples have been applied to attack speaker verification systems. Specifically, the FakeBob attacks are shown to be very effective against the state-of-the-art speaker verification systems such as GMM, i-vector, and x-vector, and can achieve more than 99% targeted attack success rate in [10]. This paper also shows that defense systems that perform well against

adversarial attacks for the images in the area of the image classification problem, such as local smoothing [11], quantization [11], and temporal dependency detection [11], cannot defeat the FakeBob attacks, which motivated our work in this thesis. One interesting attack is that it attempts to find universal adversarial perturbations, so that these same perturbations can be added to any input audio to fool a speaker verification system [38] [39].

There have been some works on the detection and defense methods against adversarial attacks in speaker verification systems. For example, a separate neural network has been proposed to detect the appearance of adversarial samples [40] [41]. Moreover, Wu et al. proposed to use voting to against adversarial examples [42]. However, the implementations of these detection or defense methods are not simple nor light-weight, and require sufficient computations. Moreover, it is not clear how these defense methods can perform against the state-of-the-art adversarial attacks such as FakeBob.

The idea of using noises to against adversarial attacks in the field of image classification has been exploited. For example, He et al. proposed to use parametric noise injection in each layer of deep neural network to enhance the robustness of an image classifier [43]. Moreover, Byun et al. appended small noise to an image before sending it to the image classifier [14]. In the introduction, we have discussed the main differences between the work in [14] and our work.

3. BACKBROUND

In this section, we provide the background of this thesis work. Specifically, we first describe the representation of the voice of a speaker in both the time domain and the Mel spectrogram. We then introduce the architecture of speaker verification (SV) systems. Finally, we present an adversarial example attack against SV systems. That is, using FakeBob [10] as an example, we introduce how the adversarial example is built and used to attack state-of-the-art SV systems.

3.1 Voice

As well known, different speakers have different voice, so that the voice can be used as biometrics to identify a person. In a computer, the voice or the audio of a speaker is represented and stored as a one-dimension array signal. Specifically, the length of the array, *n*, depends on the sampling rate (*i.e.*, numbers of samples per second) and the time duration of the voice (or audio), as shown in the following equation:

$$n = audio\ time\ duration \times sampling\ rate. \tag{3.1}$$

Then, the audio vector can be written as follows:

$$Audio\ Vector = [x_1, x_2, \dots, x_n] \tag{3.2}$$

where x_i is the amplitude of the signal at the sample i ($1 \leq i \leq n$) and reflects the strength of the signal. The value of x_i is usually in the range between -1 and 1. Figure 3.1 shows an example of an audio clip on how the amplitude of the voice changes with time. In this example, the audio time duration is 8.94 seconds, and the sampling rate is 16 kHz or 16,000 samples per second. Thus, $n = 143{,}040$. Moreover, in this example, a female speaker said in English, "I will endeavor in my statement to avoid such terms as would serve to limit the events to any particular place or give a clue as to the people concerned."

To abstract the features of a speaker's voice, the audio clip array in Equation (3.2) is usually translated into a Mel spectrogram through the transformation method called the Mel-frequency

cepstrum (MFC) [12]. In a Mel spectrogram, three dimensions are used and include the time, the frequency, and the magnitude. Figure 3.2 shows the Mel spectrogram of the audio clip shown in Figure 3.1. In the figure, the x-axis is the time, the y-axis is the frequency, and the color represents the magnitude or the strength. It can be seen that the Mel spectrogram can provide the detailed information on when and which frequency the voice has the high or low magnitude. Moreover, it is noted that the y-axis (*i.e.*, frequency) uses log-scale, instead of the linear-scale, and that the magnitude is converted to decibels (dBs), because of the characteristics of human hearing, *i.e.*, that only a very small and concentrated range of frequencies and amplitudes can be perceived. Many state-of-the-art SV systems have included a module that converts the input audio data into the Mel spectrogram as input data preprocessing.

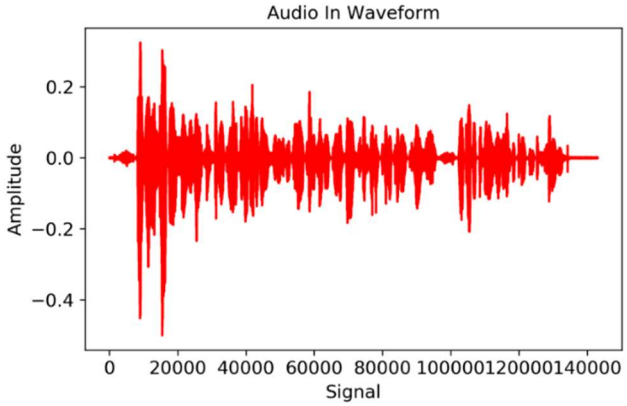


Figure 3.1 *An Audio Clip in the Time Domain*

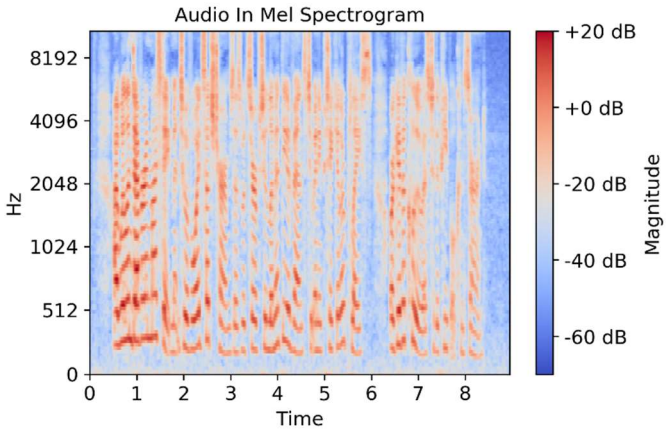


Figure 3.2 *An Audio Clip in a Mel Spectrogram*

3.2 Speaker Verification Systems

Speaker verification (SV) systems have been widely used to identify a person through their voice. In our work, we focus the score-based SV systems that most state-of-the-art SV systems belong to. In a score-based SV system, as show in Figure 3.3, it contains two phases: speaker enrollment and speaker recognition.

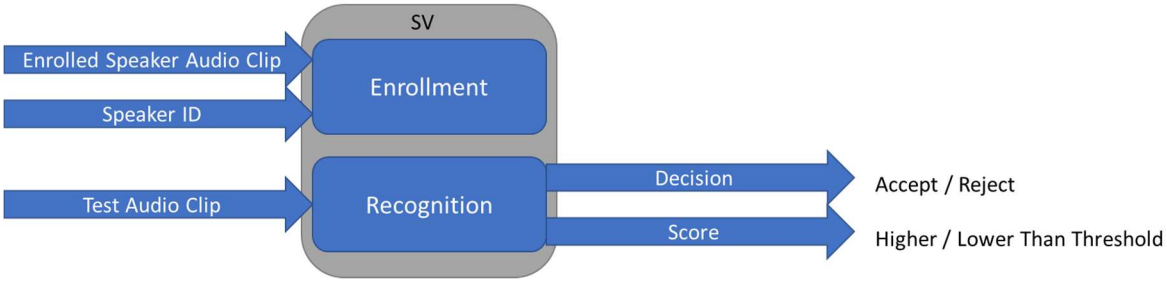


Figure 3.3 *A Score-Based Speaker Verification System*

For the phase of speaker enrollment, a speaker needs to provide the identifier and their audio clips. The SV system transfers the speaker's voice into a fixed length low dimensional vector called speaker embedding. Basically, the speaker embedding represents the features of a speaker's voice and is used to calculate the similarity between two audio clips. Different SV systems use different approaches to obtain speaker embedding. The popular SV systems include i-vector [30], GMM (Gaussian Mixture Model) [29], d-vector [31], and x-vector [32]. In this work, we focus on GMM and i-vector, since they have been widely used in real life and been applied as the baseline for comparisons. Here, we use the notation SE_R to refer to the vector of speaker embedding of the enrolled or registered speaker.

Besides obtaining the speaker embedding, the enrollment phase attempts to find a proper threshold for this speaker. Such a threshold is a key consideration for a score-based SV system. To understand the importance of the threshold, we first look at the recognition phrase. As shown in Figure 3.3, when a test audio clip is provided to the SV system, it abstracts the speaker embedding of this audio, which is referred by SE_T. Then, the SV system calculates the similarity between vector SE_R and vector SE_T and obtains a similarity score. A higher score reflects more similar between two vectors of speaker embedding. Finally, the similarity score is compared with

the threshold. If the score is higher than or equal to the threshold, the system will accept the test audio clip and treat the speaker as the enrolled user. Otherwise, the system will reject the access of the speaker. The basic process of the recognition is summarized in Algorithm 3.1.

Algorithm 3.1 *Recognition Phase in Score-Based Speaker Verification Systems*

Input: *an audio clip, SE_R, threshold*
Output: *decision*

```
1: begin
2:     SE_T ← SV astracts speaker embedding based on input audio
3:     score ← SV calculates the similarity between SE_R and SE_T
4:     if score ≥ threshold then
5:         decision ← True
6:     else
7:         decision ← False
8:     end if
9: end
```

There are two basic false cases for a SV system: (1) accepting a speaker that is not the enrolled user, and (2) rejecting the enrolled speaker. Usually, we call the speaker that is not enrolled as the illegal speaker or illegal user. For these two cases, we use the false acceptance rate (FAR) and the false rejection rate (FRR) to measure. Specifically, FAR and FRR are defined as follows:

$$FAR = \frac{number\ of\ falsely\ accepted\ illegal\ speaker\ audio\ clips}{total\ number\ of\ all\ illegal\ speaker\ audio\ clips} \tag{3.3}$$

$$FRR = \frac{number\ of\ falsely\ rejected\ enrolled\ speaker\ audio\ clips}{total\ number\ of\ all\ enrlled\ speaker\ audio\ clips} \tag{3.4}$$

For an ideal SV system, both FAR and FRR are 0. However, in a real SV system, it is difficult to make them both 0, and they are tradeoff. That is, when one of FAR and FRR decreases, the other will increase. Intuitively, when the threshold increases, it becomes more difficult for an audio clip to be accepted. As a result, FAR will decrease while FRR will increase. A proper threshold, which is used in our work, is to use the value when FAR and FRR are equal. Conventionally, when FAR is equal to FRR, the rate is called equal error rate (EER) [33].

To find the EER and the corresponding threshold, both enrolled speaker audio clips and illegal speaker audio clips need to be provided to the enrollment phase. A SV system will calculate the similarity scores for all provided audio clips and then find the threshold that can lead to the EER. The basic process of the enrollment is summarized in Algorithm 3.2, where $score_R$ and $score_I$ are two sets of scores for enrolled and illegal audio clips, respectively.

Algorithm 3.2 *Enrollment Phase in Score-Based Speaker Verification Systems*

Input: *enrolled speaker audio clips, illegal speaker audio clips*
Output: *threshold,* SE_R

```
begin
    SE_R ← SV astracts speaker embedding based on enrolled audio clip
    foreach audio in enrolled speaker audio clips
        score_R ← SV astracts speaker embedding and calculates score
    end for
    foreach audio in illegal speaker audio clips
        score_I ← SV astracts speaker embedding and calculates score
    end for
    threshold ← SV finds threshold from score_R and score_I for EER
end
```

3.3 Adversarial Attacks on Speaker Verification Systems

As shown in previous work [37] [38] [39] [44], machine learning (including deep learning) models are particularly vulnerable to adversarial attacks. Such attacks have been widely researched in the images for the image classification problem. However, adversarial attacks and defenses have not been systematically studied in the field of SV systems yet.

In the context of a SV system, adversarial attacks attempt to let the SV system falsely accept a well-designed illegal audio, which is called an adversarial audio. Specifically, adversarial audios are original illegal clean audio with small perturbations, often barely perceptible by humans. However, such perturbations lead the SV system to falsely accept the audio. Let x be an original audio from an illegal user, p denotes the perturbation vector with the same length as x, and x' be the adversarial audio. Then,

$$x' = x + p. \tag{3.5}$$

From the view of humans, there is no or little difference between *x* and *x'*, when hearing these two audios. However, from the perspective of the SV system, it will reject *x*, but falsely accept *x'*.

To make sure that the adversarial audio is not noticed or detected by humans, the perturbations are usually very small and constrained by a perturbation threshold, *i.e.*, ε. That is,

$$p = [p_1, p_2, \dots, p_n], \quad where \ -\varepsilon < p_i < \varepsilon \text{ and } 1 \le i \le n. \tag{3.6}$$

Choosing the value of ε is an important consideration for adversarial attacks. A larger value of ε makes the attack easier to succeed, but meanwhile causes it more perceptible by humans.

In our work, we study the FakeBob attack [10] and how to defend against it, since it is the state-of-the-art adversarial attack against SV systems including GMM, i-vector, and x-vector.

3.3.1 FakeBob Attack System

FakeBob attacks are a block-box attack and do not need to know the internal structure of a SV system. Moreover, as shown in [10], FakeBob achieves at least 99% targeted attack success rate (ASR) on both open source and commercial SV systems. Note that ASR can be defined as the following:

$$ASR = \frac{number\ of\ falsely\ accepted\ adversarial\ audios}{total\ number\ of\ all\ adversarial\ audios} \tag{3.7}$$

Figure 3.4 shows the basic process of FakeBob adversarial attacks.

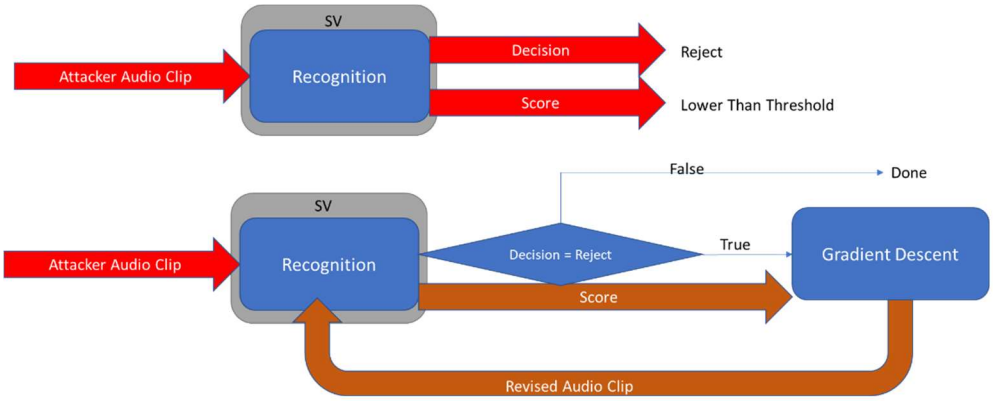


Figure 3.4 *The Process of FakeBob Adversarial Attacks*

Basically, FakeBob applies the basic iterative method (BIM) [45] and the natural evolution strategy (NES) [46] to generate the adversarial audio. That is, the attack takes multiple iterations to get the final adversarial audio (*e.g.*, x) that minimizes the following loss function or objective function

$$L(x) = max\{\theta - S(x), 0\}, \tag{3.8}$$

where θ is the threshold of the SV system and $S(x)$ is the score function that calculates the score of an input audio for SV. FakeBob attempts to resolve the optimization problem by estimating the threshold θ and finding iteratively the input audio that reduces $L(x)$, through the method of gradient decent over the input audio. That is, it applies the following gradient decent function

$$f_G(x) = \nabla_x L(x). \tag{3.9}$$

Note that here the gradient decent is different from the back-propagation widely used in deep learning, and the differentiation is based on input audios, instead of the weights of the machine learning model.

Define a sign function $f_{sign}(y)$ in the following way: For each element (*i.e.*, y_i) in the vector y, a sign function gets the sign of the value of each element in the vector, *e.g.*,

$$f_{sign}(y_i) = \begin{cases} 1, & if\ y_i > 0 \\ 0, & if\ y_i = 0 \\ -1, & if\ y_i < 0 \end{cases} \tag{3.10}$$

Moreover, assume x_i $(1 \le i \le n)$ is a signal in the original clean audio (*i.e.*, x) from an illegal speaker, x_i^m $(1 \le i \le n)$ is the corresponding signal in the adversarial audio at m^{th} iteration (*i.e.*, x^m), and ε is the perturbation threshold shown in Equation (3.6). Based on the assumption in Equation (3.6), a clip function is defined as follows

$$f_c(x_i^m) = \begin{cases} x_i^m, & if\ |x_i^m - x_i| < \varepsilon \\ x_i + \varepsilon, & if\ x_i^m \geq x_i + \varepsilon \\ x_i - \varepsilon, & if\ x_i^m \leq x_i - \varepsilon \end{cases} \tag{3.11}$$

Using the above functions, FakeBob updates the input audio through the following iteration

$$x^m = f_c(x^{m-1} - lr \times f_{sign}(f_G(x^{m-1}))) \tag{3.12}$$

where *lr* is the learning rate. The FakeBob attack is summarized in Algorithm 3.3.

Algorithm 3.3 *FakeBob Attacks*

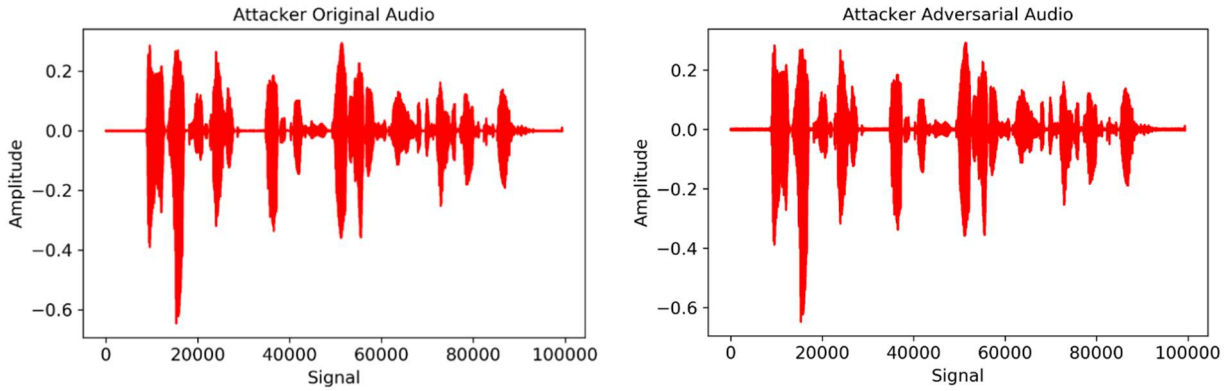
Input: *an audio signal array, threshold of target SV system*
Output: *an adversarial audio*
Require:
Threshold of target SV system $\boldsymbol{\theta}$
Audio signal array $\boldsymbol{A}$
Maximum iteration $\boldsymbol{m}$
Score function $\boldsymbol{S}$
Gradient decent function $\boldsymbol{f_G}$
Clip function $\boldsymbol{f_c}$
Learning rate $\boldsymbol{lr}$
Sign function $\boldsymbol{f_{sign}}$

```
begin
    adver ← A
    for i = 0; i < m; i++:
        score ← S(adver)
        if score ≥ θ:
            return adver
        end if
        adver ← f_c(adver − lr × f_sign(f_G(adver)))
    end for
end
```

3.3.2 Inspection of an Adversarial Audio

To better understand the FakeBob attack, we look into one example of an adversarial audio in both the time domain and the Mel spectrogram. Specifically, we registered the voice of the speaker that is used in Section 3.1 in a GMM SV system. Then we chose another speaker as an

illegal user and used a clean audio from this speaker. This audio is that a female speaker said “there chap you've got it lapped Percy while the others screamed at the sight of Jasper space…” It was verified that the audio is rejected by the GMM SV system. The time waveform of the audio is shown in Figure 3.5(a). Note that the audio is 6.21 seconds long and contains 99,360 signals under the sample rate of 16 KHz.



(a) Original Audio (b) Adversarial Audio

Figure 3.5 *Original and Adversarial Audios in the Time Domain*

Applying the FakeBob with the perturbation threshold of 0.002, we obtained the adversarial audio that is based on the illegal audio from Figure 3.5(a) and is falsely accepted the GMM SV system. The time waveform of the adversarial audio is shown in Figure 3.5(b). It can be seen that the adversarial audio is very similar to the original audio. We further plotted the perturbation vector (*i.e.*, p in Equation (3.5)) in Figure 3.6 and found that the values of perturbations are very small and are between -0.002 and 0.002. Moreover, the mean and the standard deviation of perturbations are 1.15×10^{-6} and 0.00176, respectively. Note that the standard deviation is close to the perturbation threshold.

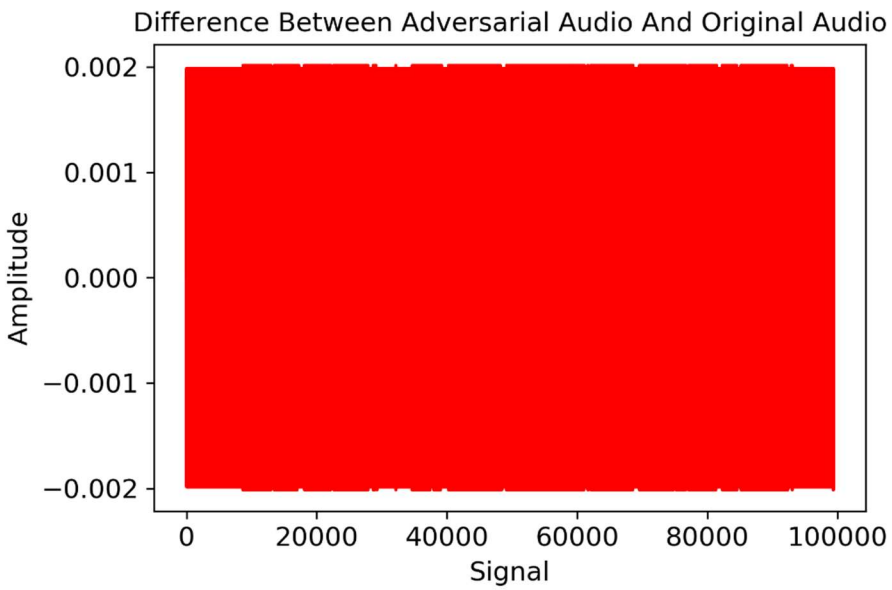


Figure 3.6 *Differences between Original and Adversarial Audios in the Time Domain*

We plot the Mel spectrograms of the original audio and the adversarial audio in Figure 3.7. It can be seen that for the blue parts in the figures, the color in the adversarial audio is whiter than that in the original audio. This indicates that there are higher magnitudes or energies in the background of the adversarial audio. We further plot the Mel spectrogram of the perturbation vector (*i.e.*, p in Equation (3.5)) in Figure 3.8. It can be seen that the perturbations are more like small white noises. Moreover, from both Figures 3.6 and 3.8, it can be seen that there is no specific pattern in both the time domain and the Mel spectrogram.

In summary, the perturbations used in FackBob attacks are very small and are similar to white noises. On the other hand, these perturbations are not random, but are intentionally designed to fool the SV systems. These observations can be applied to the defenses against FakeBob, as we will discuss in the next section.

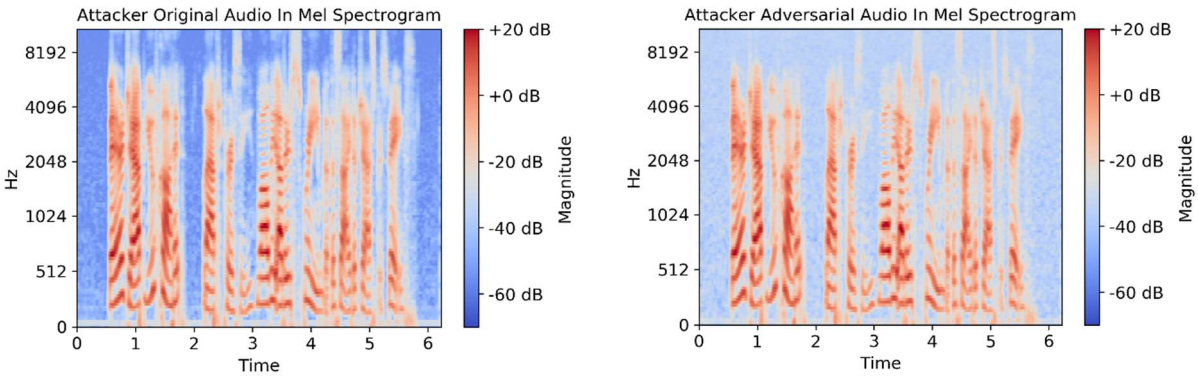


Figure 3.7 *Original and Adversarial Audios in a Mel Spectrogram*

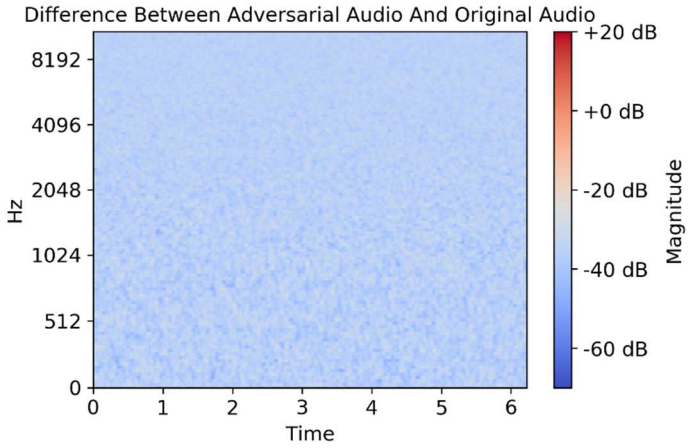


Figure 3.8 *Differences between Original and Adversarial Audios in a Mel Spectrogram*

4. PROPOSED DEFENSE SYSTEM

In this section, we propose a defense system against adversarial attacks for SV systems. Specifically, we first introduce the design goals of a defense system. Next, we describe our proposed defense system based on the observations from adversarial audios. Finally, we provide the details of the implementations of the defense system in two different approaches: denoising that attempts to remove the perturbations in adversarial audios and noise-adding that attempts to perturb adversarial audios.

4.1 Design Goals of a Defense System

To counteract the adversarial attacks in a SV system, we attempt to design and implement a defense system that achieves the following goals:

- **Simplicity**. The defense system is easy to implement and can be compatible with an existing SV system. That is, it does not require any change to the internal structure of the currently used SV system.
- **Light weight**. It does not significantly increase the computation load of the SV system. The defense method only slightly increases the processing time for an input audio.
- **Effectiveness**. The defense algorithm should be able to greatly increase the time for an attacker to generate a successful adversarial audio or significantly reduce the attack success rate of adversarial attacks such as FakeBob. On the other hand, the defense method should only slightly affect the normal operations of a SV system, such as FAR and FRR.

4.2 A Defense System

To achieve these goals, we design a defense system based on the observations from an adversarial audio. Specifically, as shown in Section 3.3, the adversarial audio is simply the clean illegal audio with well-designed perturbations that are similar to white noises. If such perturbations can be removed or modified, adversarial attacks would loss the efficiency against the SV system. Based on this intuition, we propose a defense system as shown in Figure 4.1. Comparing Figure 4.1 with Figure 3.3, it can be seen that we add an additional module, *i.e.*, plugin function, before the recognition module to a SV system. Such a plugin function is used to preprocess an input audio

to either forcefully remove the perturbations or intentionally modify the perturbations. As a result, in the following sections we will discuss two options for the plugin function: denoising and noise-adding.

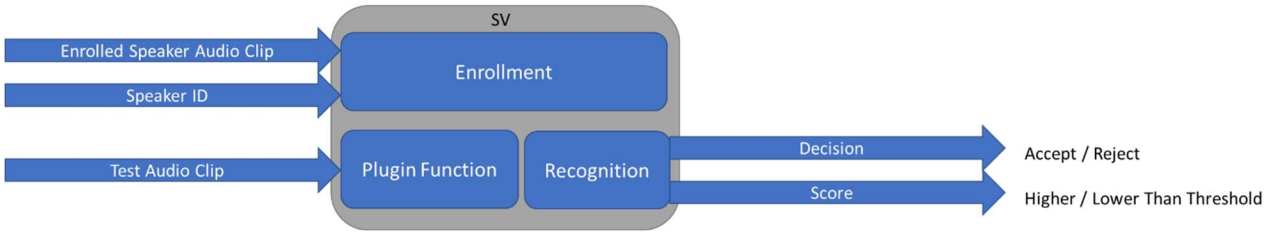


Figure 4.1 *The Proposed Defense System*

It can be seen that the proposed defense system is compatible with an existing SV system and can be applied to any SV system. It does not require to change the internal structure of a SV system. Moreover, the overhead of the defense method is only on the plugin function. If such a plugin is light weight, it will not introduce much additional computation to the SV system. Furthermore, the main goal of the plugin function is to modify the input audio so that it would have a major impact on adversarial audios, but have a minor impact on normal audios. As our first attempt, we study denosing and noise-adding functions in this work. But other functions can be applied as well, which is our future work.

4.3 Denoising

The basic idea of the denoising function is to remove or reduce the perturbations or noises in input audios. In our work, we applied the method of noise reducing proposed in [13] as our denoising function.

4.3.1 Denoising Algorithm

As indicated in [13], the denoising method uses the spectral gating to reduce noises in an audio. Specifically, given both signal and noise audio clips, it transforms time-domain waveforms into the frequency domain, then removes the noise from the signal in the frequency domain, and finally transforms the modified signal from the frequency domain back to the time domain.

To get a noise audio clip, we assume that the noise or the perturbation is white and follows the normal probability distribution, based on the observations of adversarial audios from Section 3.3. That is, we consider that the perturbations are Gaussian noise, and p_i in Equation (3.6) is assumed to have the following normal distribution with the mean of 0 and the standard deviation of σ:

$$p_i \sim N(0, \sigma^2). \tag{4.1}$$

The larger value of σ corresponds to the larger value of the perturbation threshold (*i.e.*, ε).

The transformation of a signal from the time domain to the frequency domain is based on the short-time Fourier transform (STFT), which is widely applied in digital signal processing (DSP). Similarly, inverse STFT (iSTFT) is used to transform the signal from the frequency domain back into the time domain. The algorithm of the denoising function is summarized in Algorithm 4.1.

Algorithm 4.1 *Denoise Function*

Input: *an audio clip A_1, noise variance* $\boldsymbol{\sigma^2}$
Output: *a denoised audio clip A_2*
Require Functions:
Normal distribution generator $\boldsymbol{N}$
Short-time Fourier Transform STFT
Inverse Short-time Fourier Transform iSTFT

1: $begin$
2: $noise\ clip \leftarrow [0,0,\dots,0]$
3: $for\ each\ signal\ noise[i]\ in\ noise\ clip$:
4: $noise[i] \leftarrow N(0, \sigma^2)$
5: $end\ for$
6: $n\_stft \leftarrow STFT(noise\ clip)$
7: $A_1\_stft \leftarrow STFT(A_1)$
8: $A_2\_stft \leftarrow$ *Remove noise from* A_1 *based on* n_stft *and* A_1_stft
9: $A_2 \leftarrow iSTFT(A_2\_stft)$
10: end

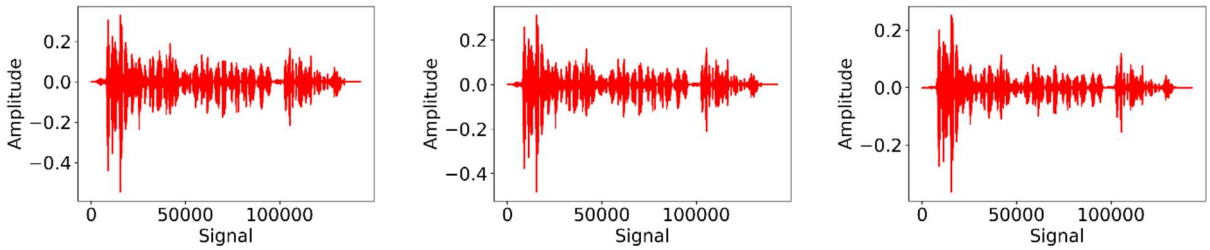
4.3.2 Inspection of the Denoising Function on a Clean Audio

Intuitively, the denoising function can reduce white-like perturbations in an audio. However, it is not clear how it can affect a clean audio. In this section, we study the effect of the denoising

function on a clean audio through inspecting denoised audios in both the time domain and the Mel spectrogram.

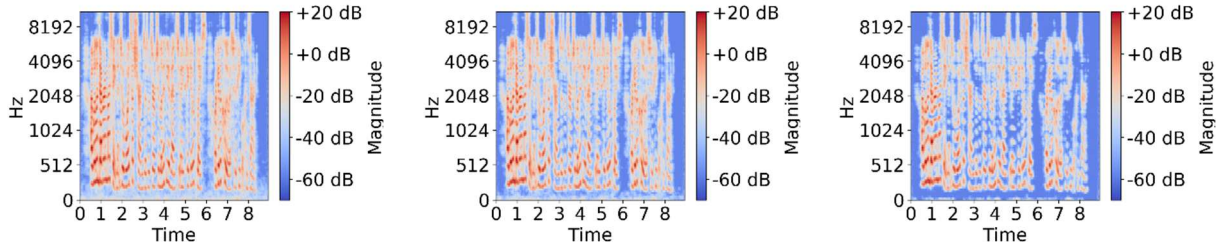
Here we choose the audio used in Section 3.1 as the clean input audio. Then, we let this audio go through the denoising function with three different standard deviations of noises, *i.e.*, $\sigma = 0.001, 0.002, and\ 0.005$. We plot the time-domain waveforms of three denoised audios in Figure 4.2. It can be seen that in these three cases, all waveforms look similar. We further plot the Mel spectrograms of three denoised audios in Figure 4.3. It can be seen that when σ increases, the background blue color becomes darker, indicating that the background noises are removed more.

While reducing the noise, the denoising function can also negatively affect the original sound. To see this, we plot the differences between the original audio and the denoised audio in both the time domain and the Mel spectrogram in Figures 4.4 and 4.5, respectively. As shown in time-domain waveforms in Figure 4.4, it can be seen that the differences are in the similar waveforms as the original audio. Moreover, when σ increases, the amplitude of the differences increases as well. In Figures 4.3 and 4.5, it shows that the voice part (*i.e.*, orange color) becomes darker as σ increases. This indicates that the denoised function intentionally enhances the original sound, which can cause the "echo" effect on an input audio. When we heard these denoised audios, we did notice such echo. As σ increases, the echo is stronger. In the next chapter, we will further show how the denoising function affects the normal operations of a SV system, such as FAR, FRR, and EER.



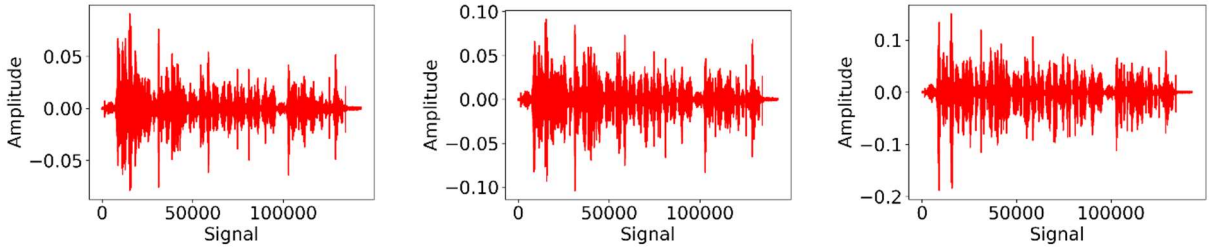
(a) σ = 0.001 (b) σ = 0.002 (c) σ = 0.005

Figure 4.2 *Denoised Audios in the Time Domain*



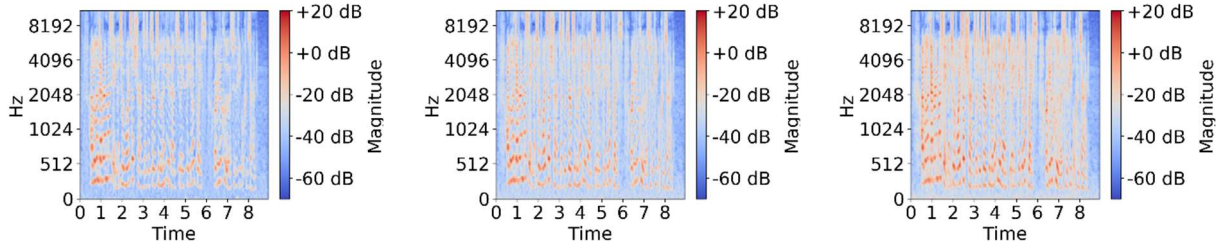
(a) $\sigma = 0.001$ (b) $\sigma = 0.002$ (c) $\sigma = 0.005$

Figure 4.3 *Denoised Audios in a Mel Spectrogram*



(a) $\sigma = 0.001$ (b) $\sigma = 0.002$ (c) $\sigma = 0.005$

Figure 4.4 *Differences between Original and Denoised Audios in the Time Domain*



(a) $\sigma = 0.001$ (b) $\sigma = 0.002$ (c) $\sigma = 0.005$

Figure 4.5 *Differences between Original and Denoised Audios in a Mel Spectrogram*

4.4 Noise-Adding

Different from the denoising function, the noise-adding function attempt to perturb adversarial audios so that adversarial attacks would loss or reduce the ability to mislead a SV system.

4.4.1 Noise-Adding Algorithm

The noise-adding function appends some small noises to the input audio. Noises can be many different options. As a first attempt, we apply Gaussian noises with the mean of 0 and the standard deviation of σ. That is,

$$A_2 = A_1 + Noise \tag{4.2}$$

where A_1 is the input audio, A_2 is the noise-added audio, and

$$each\ element\ in\ Noise \sim N(0, \sigma^2). \tag{4.1}$$

The algorithm of the noise-adding function is shown in Algorithm 4.2.

Algorithm 4.2 *Noise-Adding Function*

Input: *an audio clip A_1, noise variance* $\boldsymbol{\sigma^2}$
Output: *a noise-added audio clip A_2*
Require Function: *Normal distribution generator* ***N***

```
begin
    noise clip ← (0,0, ...,0), size of the array = length(A_1)
    for each signal noise[i] in noise clip:
        noise[i] ← N(0, σ^2)
    end for
    A_2 ← A_1 + noise clip
end
```

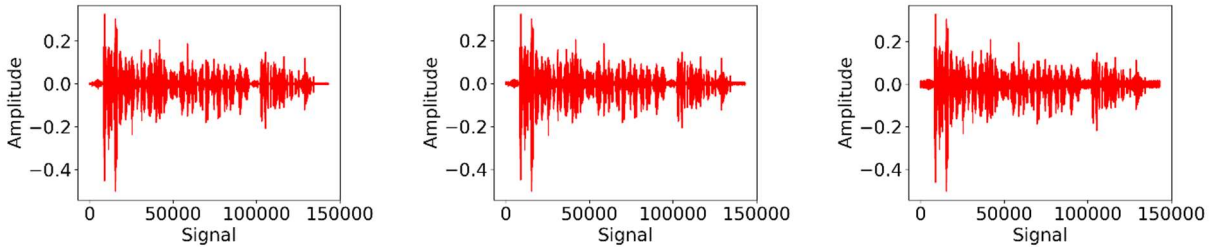
4.4.2 Inspection of the Noise-Adding Function on a Clean Audio

The noise-adding function is a very simple method and intends to make well-design perturbations into more random perturbations so that the adversarial attacks have less chance to succeed. In this section, we provide our intuitions and reasoning why noise-adding can work against adversarial attacks. Moreover, similar to Section 4.3.2, we study the effect of the noise-adding function on a clean audio through inspecting noise-added audios in both the time domain and the Mel spectrogram.

Intuitively, noise-adding can provide the certain protection against adversarial attacks for the following reasons: (1) Many adversarial attacks, such as FakeBob, apply gradient decent to generate adversarial examples, as shown in Section 3.3. Adding small noises to input audios, however, can modify or distort the mapping between input audios and the corresponding output scores. That is, under the noise-adding plugin, the output score is no longer accurately reflect the original input audio, so that it becomes more difficult for the gradient decent method to change the input audio to increase the output score in a SV system. As shown in our experiments in Section 5, the FakeBob takes much longer time to generate adversarial audios in a SV system with noise-adding than in a SV system without it. (2) Although perturbations are similar to white noises, they are not random and are well designed to fool a SV system. Noise-adding can potentially introduce randomness into perturbation to destroy or modify such well-designed perturbations.

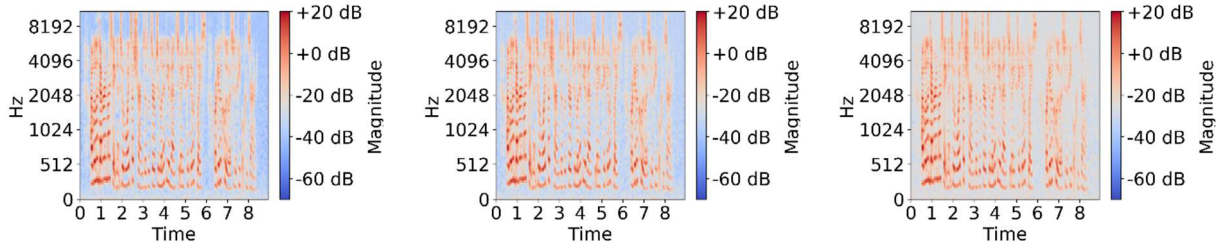
We use the same audio as that applied in Sections 3.1 and 4.3.2 as the clean input audio, and employ the noise-adding function to it. Specifically, three cases are considered with using different standard deviations for the noise, *i.e.*, $\sigma = 0.001, 0.002, and\ 0.005$. We plot the time-domain waveforms of three noise-added audios in Figure 4.6. It can be seen that in these three cases, all waveforms look similar. We further plot the Mel spectrograms of three noise-added audios in Figure 4.7. It can be seen that when σ increases, the background blue color becomes lighter, indicating that the background noises are added more. Comparing Figures 4.3 and 4.7, it is interesting to see that denoising and noise-adding are two opposite operations. While denoising attempts to reduce noises, noise-adding introduces more noises.

Figures 4.8 and 4.9 show the differences between the original audio and the noise-added audio in the time domain and the Mel spectrogram, respectively. As expected, when σ increases, the amplitude of the differences becomes larger in the time domain, and the color of the differences becomes lighter (*i.e.*, more energy for the background noises) in the Mel spectrogram. In the next chapter, we will further show how the noise-adding function affects the normal operations of a SV system, such as FAR, FRR, and EER.



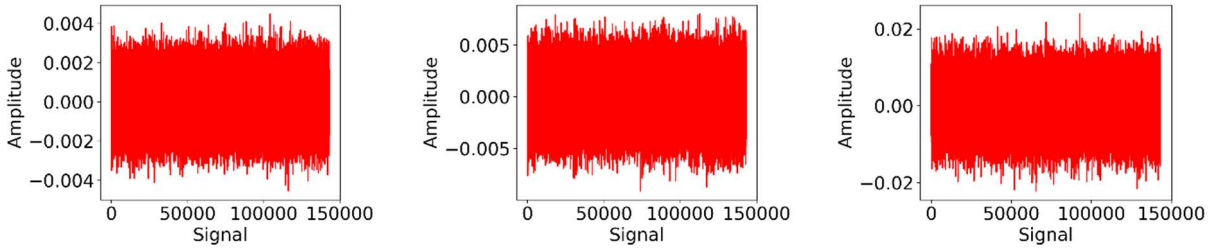
(a) $\sigma = 0.001$ (b) $\sigma = 0.002$ (c) $\sigma = 0.005$

Figure 4.6 *Noise-Added Audios in the Time Domain*



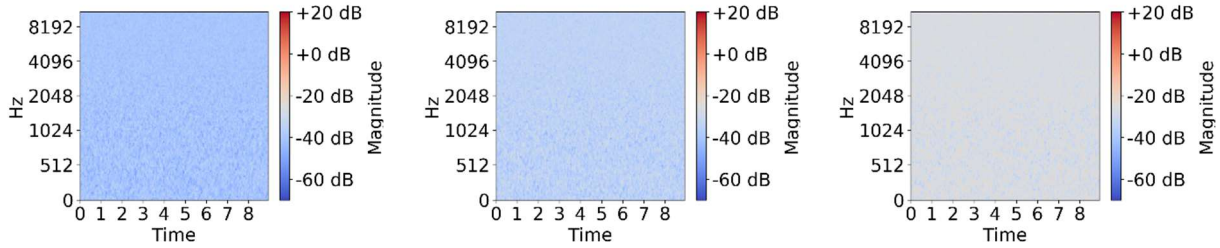
(a) $\sigma = 0.001$ (b) $\sigma = 0.002$ (c) $\sigma = 0.005$

Figure 4.7 *Noise-Added Audios in a Mel Spectrogram*



(a) $\sigma = 0.001$ (b) $\sigma = 0.002$ (c) $\sigma = 0.005$

Figure 4.8 *Differences between Original and Noise-Added Audios in the Time Domain*



(a) $\sigma = 0.001$ (b) $\sigma = 0.002$ (c) $\sigma = 0.005$

Figure 4.9 *Differences between Original and Noise-Added Audios in a Mel Spectrogram*

5. PERFORMANCE EVALUATIONS

In this section, we evaluate the effectiveness and the efficiency of the proposed defense system against adversarial attacks in SV systems. Specifically, we first describe the experimental setup. We then measure the impact of the defense methods on the normal operations of SV systems. Finally, we evaluate the performance of our proposed defense system against the state-of-the-art adversarial attacks, i.e., FakeBob attacks.

5.1 Experimental Setup

The experiments were run parallelly in two virtual machines (VMs) provided by Google Cloud Platform [47] and a local GPU server. Two VMs are with 8-core Intel Xeon CPU 3.1 GHz and 32 GB memory, whereas the GPU server is with 24-core Intel I9 CPU 2.9 GHz, 128 GB memory and GeForce RTX 2080 Ti graphic card. All machines are installed with Ubuntu 20.04.

Our experiments focus on GMM and i-vector SV systems. These SV systems are implemented by Kaldi speech recognition toolkit [48] and use the pre-trained models from VoxCelab 1 [49]. Moreover, the adversarial attacks against these SV systems are implemented through FakeBob attacks [10] with the perturbation threshold of 0.002 (*i.e.*, $\varepsilon = 0.002$) and 1,000 maximum iterations (*i.e.*, m = 1, 000).

The audio dataset used is from LibriSpeech [50] and contains 8 different speakers, which were also applied in FakeBob [10]. For each speaker, there are 100 audio clips, each of which lasts between 4 seconds to 10 seconds.

It is noted that a SV system, either GMM or i-vector, without the proposed defense system, is deterministic. That is, when an input audio is accepted (or rejected) by the SV system, it will be always accepted (or rejected) for future testing in the same SV system. However, a SV system that is installed with our proposed defense system becomes stochastic, because of the effect of the Gaussian noise generation. That is, if an input audio is accepted (or rejected) in the current testing for the SV system, it may be rejected (or accepted) next time when it is fed into the same SV system. To count for such a stochastic effect, during the testing we let an input audio repeat going through a SV system for 100 times to get the average of the performance metrics such as EER and ASR, which we have introduced in Section 3.

5.2 Performance Evaluations of the Proposed Defense System for Normal Operations of Speaker Verification Systems

In our experiments, we first study the impact of the proposed defense methods, *i.e.*, denoising and noise-adding, on the normal operations of a SV system. Specifically, we use EER as the performance metric here. Recall that EER is the rate of FAR or FRR when they are equal, and FAR and FRR are defined in Equations (3.3) and (3.4), respectively. A smaller EER reflects a better SV system. Moreover, the added plugin function to a SV system would slow down the processing of an input audio. As a result, we also record the processing time in our experiments.

Among 8 speakers provided, we select two speakers with the identifier of 61 and 2830 as registered users in two separate experiments. As shown in Section 5.3, the FakeBob attacks against these two users can achieve 100% ASR in a SV system without our proposed defense system. To obtain the EER, 100 audio clips from a registered speaker were tested, whereas another 100 audio clips were randomly chosen from all illegal speakers to be fed into the SV system.

Table 5.1 shows the performance of a GMM SV system with or without the denoising or noise-adding function for speakers 2830 and 61, as well as with different values of the standard deviation of noises (*i.e.*, $\sigma = 0.001, 0.002, and\ 0.005$). Note that the original GMM SV system, which is without the defense plugin, can be regarded as the special case when $\sigma = 0$ for either denoising or noise-adding. It can be seen that for most cases, when σ increases, EER increases for both denoising and noise-adding. However, the value of EER only increase slightly, especially when σ is not large; for example, $\sigma \leq 0.002$. As a reference, in Table 5.1 we also record the threshold of the SV system (*i.e.*, θ). Such a threshold will be applied for studying adversarial attacks in Section 5.3. It can be seen that when σ increases, the threshold decreases in general.

It can also be seen from Table 5.1 that in a GMM SV system, the denoising function takes much longer time than the noise-adding function to process all 200 testing audios. The overhead of the noise-adding is very light, because the plugin of adding random Guassian noises does not require too much computation. On the other hand, the denoising function needs to apply both STFT and iSTFT operations, which demand a lot of computation.

We further summarize the performance of the proposed defense system for normal operations of an i-vector SV system in Table 5.2. Similar to results in Table 5.1, it can be seen that EER does not increase too much when σ increases, especially when $\sigma = 0.001\ or\ 0.002$. Different from results in Table 5.1, the processing time of an i-vector SV system with the defense is more similar

to that without the defense. The reason is that the i-vector SV system requires longer time to test all 200 audios than the GMM SV system, but the overhead of a plugin is fixed.

Table 5.1 *Performance Evaluations of the Proposed Defense System on Normal Operations of GMM SV systems*

Speaker	Plugin	σ	EER (%)	Threshold	Processing Time
2830	original	0	1.12	0.0610	17.58
2830	denoising	0.001	2.25	0.0722	30.91
2830	denoising	0.002	3.37	0.0708	30.53
2830	denoising	0.005	2.92	0.0501	31.47
2830	noise-adding	0.001	1.35	0.0770	18.77
2830	noise-adding	0.002	2.58	0.0454	19.16
2830	noise-adding	0.005	5.84	-0.0304	19.50
61	original	0	0.97	0.1285	19.29
61	denoising	0.001	0.97	0.1215	30.43
61	denoising	0.002	2.52	0.1369	30.29
61	denoising	0.005	3.79	0.0911	30.10
61	noise-adding	0.001	1.07	0.1098	19.91
61	noise-adding	0.002	1.26	0.0810	20.39
61	noise-adding	0.005	2.04	0.0270	21.11

Table 5.2 *Performance Evaluations of the Proposed Defense System on Normal Operations of i-Vector SV systems*

Speaker	Plugin	σ	EER (%)	Threshold	Processing Time
2830	original	0	0.0	2.1406	390.42
2830	denoising	0.001	0.0	1.7603	406.68
2830	denoising	0.002	0.0	1.4734	404.92
2830	denoising	0.005	0.0	1.5418	404.92
2830	noise-adding	0.001	0.0	2.1523	393.14
2830	noise-adding	0.002	0.0	1.9582	394.13
2830	noise-adding	0.005	0.34	1.5960	394.12
61	original	0	0.0	2.1077	476.48
61	denoising	0.001	0.29	1.9156	488.05
61	denoising	0.002	0.1	1.8095	490.72
61	denoising	0.005	0.97	1.9277	487.47
61	noise-adding	0.001	0.87	2.0787	477.55
61	noise-adding	0.002	0.78	1.8908	477.64
61	noise-adding	0.005	1.94	1.7152	476.89

From both Tables 5.1 and 5.2, we can conclude that the proposed defense system only slightly degrades the performance of a SV system, especially when σ is small. Moreover, in a GMM SV system, the noise-adding function is preferred to the denoising function, because of the much better processing time. The additional processing time of noise-adding is minor in both GMM and i-vector SV systems.

5.3 Performance Evaluations of the Proposed Defense System against FakeBob Attacks

Next, we study the performance of our proposed defense methods against FackBob attacks in a SV system. Specifically, we use ASR as the performance metric, which definition can be found from Equation (3.7). A smaller value of ASR indicates a better defense performance. Moreover, FakeBob uses multiple iterations to create an adversarial audio, as shown in Algorithm 3.3. Therefore, we also measure the average of the numbers of iterations and the average running time for FakeBob attacks to find an adversarial audio in our experiments. A larger number of the average iterations and a longer average running time reflect a better defense system against adversarial attacks.

For adversarial attacks, we randomly selected 5 audio clips from each of four illegal speakers. Thus, 20 cases of adversarial attacks were studied to obtain the average ASR, the average number of iterations, and the average running time.

Table 5.3 shows the experimental results of our proposed defense functions, *i.e.*, denoising and noise-adding, against FakeBob attacks in a GMM SV system for speakers 2830 and 61, when $\sigma = 0, 0.001, 0.002, and\ 0.005$, and a i-vector SV system for speaker 2830 and 61, when $\sigma = 0\ and\ 0.002$. It can be seen that while FakeBob achieves 100% ASR for the original GMM SV system (*i.e.*, $\sigma = 0$), the ASR of the SV system with the defense is less than 100%. Moreover, when σ increases, ASR decreases in general. For example, when $\sigma = 0.002$, ASR is averagely $(64.6\% + 47.5\%)/2 = 56.05\%$ for the denoising function, whereas it is only $(6.8\% + 3.6\%)/2 = 5.2\%$ for the noise-adding method. Moreover, FakeBob achieves $(90\% + 100\%)/2 = 95\%$ ASR for the original i-vector SV system, but the ASRs of the SV system with the proposed defense is less than 95%. When $\sigma = 0.002$, ASR is averagely $(36.8\% + 40.5\%)/2 = 38.65\%$ for the denoising function, whereas it is only $(0\% + 1\%)/2 = 0.5\%$ for the noise-adding method. It can be clearly seen that noise-adding performs much better than denoising based on the ASR. Moreover, noise-adding leads to much larger value of the average number of iterations

Table 5.3 *Performance Evaluations of the Proposed Defense System against FakeBob Attacks*

Speaker	Plugin	σ	EER (%)	Avg iter	Avg time	Avg ASR
Model	GMM					
2830	original	0	1.12	13.5	113.54	100%
2830	denoising	0.001	2.25	13.4	131.55	79.0%
2830	denoising	0.002	3.37	12.1	128.85	64.6%
2830	denoising	0.005	2.92	18.1	186.46	50.7%
2830	noise-adding	0.001	1.35	67.0	388.53	23.2%
2830	noise-adding	0.002	2.58	634.9	4423.60	6.8%
2830	noise-adding	0.005	5.84	701.8	3925.76	6.8%
61	original	0	0.97	32.5	203.82	100%
61	denoising	0.001	0.97	24.4	252.48	75.4%
61	denoising	0.002	2.52	33.6	343.07	47.5%
61	denoising	0.005	3.79	26.5	285.09	51.3%
61	noise-adding	0.001	1.07	118.2	841.30	25.5%
61	noise-adding	0.002	1.26	575.0	3562.15	3.6%
61	noise-adding	0.005	2.04	688.1	4774.93	1.4%
Model	I-Vector					
2830	original	0	0.00	261.4	9384.32	90.0%
2830	denoising	0.002	0.00	143.7	5421.76	36.8%
2830	noise-adding	0.002	0.00	958.7	34460.64	0.0%
61	original	0	0.00	76.4	2776.62	100%
61	denoising	0.002	0.10	57.5	2228.28	40.5%
61	noise-adding	0.002	0.78	877.8	31573.95	1.0%

and much longer average running time than denoising and the original SV system. The average number of iterations and the average running time for denoising are similar to those in the original SV system. However, with the noise-adding plugin, the values of these two metrics are much larger. For example, in the original GMM SV system, the average of the average number of iterations is $(13.5 + 32.5)/2 = 23$, and the average of the average running time is $(113.54 + 203.82)/2 = 158.68$. But in the defense system with noise-adding and $\sigma = 0.002$, the averages of the average number of iterations and the average running time are $(634.9 + 575.0)/2 = 604.95$ and $(4423.60 + 3562.15)/2 = 3992.875$, respectively. It indicates that noise-adding slows down the processing more than 25 times and makes FakeBob significantly harder to find the adversarial audios in GMM SV systems. Moreover, it shows a similar result for the i-vector SV system from Table 5.3. In the original i-vector SV system, the average of the average number of iterations is $(261.4 + 76.4)/2 = 168.9$, and the average of the average running time is $(9384.32 +$

$2776.62)/2 = 6080.47$. But in the defense system with noise-adding and $\sigma = 0.002$, the averages of the average number of iterations and the average running time are $(958.7 + 877.8)/2 = 918.25$ and $(34460.64 + 31573.95)/2 = 33017.295$, respectively. In the case of an i-vector SV system with the noise-adding plugin, there are only 4 cases getting successful adversarial audios among overall 40 cases. Other attacking cases reach the preset maximal iteration of 1000. Therefore, noise-adding slows down the processing more than 5 times and makes FakeBob harder to find adversarial audios in the i-vector SV system.

In summary, we can see from the experimental results that the noise-adding defense with a reasonable value of the standard deviation of noises (*e.g.*, $\sigma = 0.002$) can effectively and efficiently counteract FakeBob attacks. Moreover, such a defense method is simple to implement, is very light-weight, and only degrades the performance of normal operations of a SV system slightly.

6. CONCLUSIONS AND FUTURE WORK

In this work, we have attempted to design and implement a defense system that is simple, light-weight, and effectively against adversarial attacks in SV systems. Our designed system is based on the observations that the perturbations in an adversarial audio are very small and similar to white noises but are not random and intentionally designed to fool SV. We have proposed to add the plugin function to preprocess an input audio so that perturbations can be removed or modified to loss the effect. We have studied two opposite plugin functions, *i.e.*, denoising and noise-adding, and found that noise-adding has a much better performance against FakeBob adversarial attacks than denoising. Specifically, noise-adding with $\sigma = 0.002$ in a GMM SV systems is able to slow down the speed of FakeBob to generate adversarial audios 25 times and reduce the targeted ASR from 100% to 5.2%. Moreover, noise-adding with $\sigma = 0.002$ has a minor effect on normal operations of a SV system and has a slightly higher EER than that in the SV system without the defense. Therefore, we believe that this simple solution, i.e., noise-adding plugin, should be applied to any SV system to counteract adversarial attacks such as FakeBob.

As our on-going work, we will extend our study to other SV systems such as d-vector [31] and x-vector [32]. Moreover, we plan to research the effect of adding other different types of noises, such as Rustle noises [51], on the normal operations of a SV system and against adversarial attacks.

7. REFERENCES

[1] "Google Assistant, your own personal Google," Google, [Online]. Available: https://assistant.google.com/. [Accessed July 2021].

[2] "Siri-Apple," Apple, [Online]. Available: https://www.apple.com/siri/. [Accessed July 2021].

[3] "Amazon Alexa Voice AI | Alexa Developer Official Site," Amazon, [Online]. Available: https://developer.amazon.com/en-US/alexa. [Accessed July 2021].

[4] "Configure Cortana in Windows 10 - Configure Windows | Microsoft Docs," Microsoft, [Online]. Available: https://docs.microsoft.com/en-us/windows/configuration/cortana-at-work/cortana-at-work-overview. [Accessed July 2021].

[5] Y. B. Krishna and S. Nagendram, "ZIGBEE Based Voice Control System for Smart Home," *International Journal of Computer Technology and Applications,* vol. 03, pp. 163-168, January 2012.

[6] R. K. Das, X. Tian, T. Kinnunen and H. Li, "The Attacker's Perspective on Automatic Speaker Verification: An Overview," in *Proc. Interspeech 2020*, Shanghai, China, 2020.

[7] Z. Wu, N. Evans, T. Kinnunen, J. Yamagishi, F. Alegre and H. Li, "Spoofing and countermeasures for speaker verification: A survey," *Speech Communication,* vol. 66, no. C, pp. 130-153, February 2015.

[8] Z. Wu and H. Li, "Voice conversion versus speaker verification: an overview," *APSIPA Transactions on Signal and Information Processing,* vol. 3, December 2014.

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, Banff, Canada, 2014.

[10] G. Chen, S. Chen, L. Fan, X. Du, Z. Zhao, F. Song and Y. Liu, "Who is Real Bob? Adversarial Attacks on Speaker Recognition Systems," in *IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, 2021.

[11] Z. Yang, B. Li, P. Chen and D. Song, "Characterizing audio adversarial examples using temporal dependency," in *International Conference on Learning Representations*, New Orleans, Louisiana, USA, 2019.

[12] L. Muda, M. Begam and I. Elamvazuthi, "Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (dtw) techniques," *Journal of Computing,* vol. 2, no. 3, March 2010.

[13] T. Sainburg, "timsainb/noisereduce: v1.0," Zenodo, 2019. [Online]. Available: https://github.com/timsainb/noisereduce.

[14] J. Byun, H. Go and C. Kim, "Small Input Noise is Enough to Defend Against Query-based Black-box Attacks," *arXiv:2101.04829,* 2021.

[15] S. Arik, G. Diamos, A. Gibiansky, J. Miller, K. Peng, W. Ping, J. Raiman and Y. Zhou, "Deep Voice 2: Multi-Speaker Neural Text-to-Speech," in *Advances in Neural Information Processing Systems*, Long Beach, CA, USA, 2017.

[16] S. O. Arik, J. Chen, K. Peng, W. Ping and Y. Zhou, "Neural Voice Cloning with a Few Samples," in *Advances in Neural Information Processing Systems*, Montréal, Canada, 2018.

[17] S. Jung and H. Kim, "Neural voice cloning with a few low-quality samples," *arXiv:2006.06940,* 2020.

[18] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman and J. Miller, "Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning," in *International Conference on Learning Representations*, Vancouver, BC, Canada, 2018.

[19] Y. Jia, Y. Zhang, R. J. Weiss, Q. Wang, J. Shen, F. Ren, Z. Chen, P. Nguyen, R. Pang, I. L. Moreno and Y. Wu, "Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis," in *Advances in Neural Information Processing Systems*, Montréal, Canada, 2018.

[20] Y. Wang, R. 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 and R. A. Saurous, "Tacotron: Towards End-to-End Speech Synthesis," in *Proc. Interspeech 2017*, Stockholm, Sweden, 2017.

[21] L. Wan, Q. Wang, A. Papir and I. L. Moreno, "Generalized End-to-End Loss for Speaker Verification," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Alberta, Canada, 2018.

[22] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis and Y. Wu, "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Alberta, Canada, 2018.

[23] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," *arXiv:1609.03499,* 2016.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention Is All You Need," in *Advances in Neural Information Processing Systems*, Long Beach, CA, USA, 2017.

[25] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," in *ICASSP '83. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Boston, Massachusetts, USA, 1983.

[26] M. Morise, F. Yokomori and K. Ozawa, "WORLD: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Transactions on Information and Systems,* Vols. E99-D, pp. 1877-1884, July 2016.

[27] A. Torfi, J. Dawson and N. M. Nasrabadi, "Text-Independent Speaker Verification Using 3D Convolutional Neural Networks," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, San Diego, USA, 2018.

[28] C. Yan, Y. Long, X. Ji and W. Xu, "The Catcher in the Field: A Fieldprint based Spoofing Detection for Text-Independent Speaker Verification," in *CCS '19: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, London, United Kindom, 2019.

[29] D. A.Reynolds, T. F.Quatieri and R. B.Dunn, "Speaker Verification Using Adapted Gaussian Mixture Models," *ScienceDirect,* vol. 10, no. 1-3, pp. 19-41, January 2000.

[30] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel and P. Ouellet, "Front-End Factor Analysis for Speaker Verification," *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 19, pp. 788-798, 2011.

[31] E. Variani, X. Lei, E. McDermott, I. L. Moreno and J. Gonzalez-Dominguez, "Deep Neural Networks For Small Footprint Text-Dependent Speaker Verification," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, 2014.

[32] D. Snyder, D. Garcia-Romero, G. Sell, A. McCree, D. Povey and S. Khudanpur, "Speaker recognition for multi-speaker conversations using x-vectors," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, 2019.

[33] J.-M. Cheng and H.-C. Wang, "A method of estimating the equal error rate for automatic speaker verification," in *IEEE International Symposium on Chinese Spoken Language Processing*, Hong Kong, China, 2004.

[34] G. Heigold, I. Moreno, S. Bengio and N. Shazeer, "End-to-end text-dependent speaker verification," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, China, 2016.

[35] W. Du, Computer & Internet Security: A Hands-on Approach, Second Edition, Du Wenliang, 2019 .

[36] I. J. Goodfellow, J. Shlens and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *arXiv:1412.6572,* March 2015.

[37] X. Yuan, P. He, Q. Zhu and X. Li, "Adversarial Examples: Attacks and Defenses," *IEEE Transactions on Neural Networks and Learning Systems,* 2019.

[38] W. Zhang, S. Zhao, L. Liu, J. Li, X. Cheng, T. F. Zheng and X. Hu, "Attack on Practical Speaker Verification System Using Universal Adversarial Perturbations," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toronto, Canada, 2021.

[39] J. Li, X. Zhang, C. Jia, J. Xu, L. Zhang, Y. Wang, S. Ma and W. Gao, "Universal Adversarial Perturbations Generative Network For Speaker Recognition," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, London, United Kingdom, 2020.

[40] X. Li, N. Li, J. Zhong, X. Wu, X. Liu, D. Su, D. Yu and H. Meng, "Investigating Robustness of Adversarial Samples Detection for Automatic Speaker Verification," in *Proc. Interspeech 2020*, Shanghai, China, 2020.

[41] H. Wu, X. Li, A. T. Liu, Z. Wu, H. Meng and H.-y. Lee, "Adversarial Defense for Automatic Speaker Verification by Cascaded Self-Supervised Learning Models," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toronto, Canada, 2021.

[42] H. Wu, Y. Zhang, Z. Wu, D. Wang and H.-y. Lee, "Voting for the right answer: Adversarial defense for speaker verification," in *Proc. Interspeech 2021*, Brno, Czech Republic, 2021.

[43] Z. He, A. S. Rakin and D. Fan, "Parametric Noise Injection: Trainable Randomness to Improve Deep Neural Network Robustness Against Adversarial Attack," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019.

[44] Y. Xie, C. Shi, Z. Li, J. Liu, Y. Chen and B. Yuan, "Real-time, Universal, and Robust Adversarial Attacks Against Speaker Recognition Systems," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, 2020.

[45] A. Kurakin, I. J. Goodfellow and S. Bengio, "Adversarial examples in the physical world," in *International Conference on Learning Representations*, Toulon, France, 2017.

[46] A. Ilyas, L. Engstrom, A. Athalye and J. Lin, "Black-box adversarial attacks with limited queries and information," in *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, 2018.

[47] "Cloud Computing Services," Google, [Online]. Available: https://cloud.google.com/. [Accessed July 2021].

[48] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motl´ıcek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer and K. Vesely, "The Kaldi Speech Recognition Toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, Big Island, Hawaii, USA, 2011.

[49] A. Nagrani, J. S. Chung and A. Zisserman, "VoxCeleb2: Deep Speaker Recognition," in *Proc. Interspeech 2018*, Hyderabad, Telangana, India, 2018.

[50] V. Panayotov, G. Chen, D. Povey and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, South Brisbane, Queensland, Australia, 2015.

[51] "Rustle noise - Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Rustle_noise. [Accessed July 2021].

[52] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv:1609.04747,* 2016.

[53] S. Umesh, L. Cohen and D. J. Nelson, "Fitting the Mel scale," in *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*, Phoenix, Arizona, U.S.A, 1999.