Defending Against Adversarial Attacks in Speaker Verification Systems

Li-Chi Chang
Department of Computer Science
Purudue University Fort Wayne
Fort Wayne, IN 46805
chanl01@pfw.edu

Zesheng Chen
Department of Computer Science
Purudue University Fort Wayne
Fort Wayne, IN 46805
chenz@pfw.edu

Chao Chen
Deapartment of Electrical and Computer Engineering
Purudue University Fort Wayne
Fort Wayne, IN 46805
chenc@pfw.edu

Guoping Wang
Deapartment of Electrical and Computer Engineering
Purudue University Fort Wayne
Fort Wayne, IN 46805
wang@pfw.edu

Zhuming Bi
Department of Civil and Mechanical Engineering
Purudue University Fort Wayne
Fort Wayne, IN 46805
biz@pfw.edu

***Abstract*—In smart home, voice control becomes a main interface between users and 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 illegal access. In recent studies, however, it has been shown that speaker verification systems are particularly vulnerable to adversarial attacks. In this work, we attempt to design and implement a defense system that is simple, light-weight, and effective against adversarial attacks for speaker verification. Specifically, we study two opposite operations to preprocess input audios in speaker verification systems against adversarial attacks: 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 only has a minor effect on the normal operations of a speaker verification system.**

Keywords—speaker verification, adversarial attack, defense system, denoising, and noise-adding

I. INTRODUCTION

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. These smart home devices can be accessed and controlled through different methods, such as position detection, habit record, and most importantly, voice control. Because of its convenience and ease of operation, voice control becomes a main interface between users and 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. Compared with other verification methods, speaker verification has the advantages of being hand free and distance flexible. That is, speaker verification does not require a person to have physical 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 illegal access from attackers [1]. The main attacks include replay attacks [2], voice cloning attacks [3], and adversarial attacks [4]. Among all attacks, adversarial attacks are the most dangerous and very difficult to detect and defend [5]. 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 [5], 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 [6], quantization [6], and temporal dependency detection [6], cannot defeat the FakeBob attacks.

As a result, an important research question arises: 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 effective against adversarial attacks. Specifically, the defense system should be compatible with any 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. It is equally important that 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 [7]. 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 be less effective 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 [8]. The goal of the noise-adding is to append some small Gaussian noises to input audios to perturb adversarial audios, so that adversarial attacks would lose or reduce the ability to mislead a speaker verification system. Denoising and noise-adding are indeed opposite operations. Through experiments using the state-of-the-art speaker verification systems such as the Gaussian mixture model (GMM), we find that both methods, especially the noise-adding method, can significantly reduce the attack success rate of FakeBob.

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 [9] [10]. Moreover, Wu et al. proposed to use voting to against adversarial examples [11]. However, the implementations of these detection or defense methods are not simple nor light-weight, and require significant computations. Moreover, it is not clear how these defense methods perform against the state-of-the-art adversarial attacks such as FakeBob.

The most relevant work to our approach was recently presented in [12]. 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 [12] 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 [5], many defense systems that perform well for images cannot be applied to audios. (2) Image classification and speaker verification usually 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 GMM. The same defense mechanism may have different performance on distinct machine learning models. (3) The work in [12] aims at untargeted adversarial attacks, whereas our work focuses on targeted adversarial attacks.

We summarize our main discoveries and contributions in the following:

- We find and show that the perturbations in an adversarial audio are very small and are similar to white noises, using the time domain waveform and the Mel spectrogram. 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 effective against adversarial attacks in speaker verification systems.
- We find that the denoising function is able to reduce or remove the perturbations. As shown in our experiments based on FakeBob [5] against a GMM speaker verification system, denoising can reduce the targeted attack success rate from 100% to 56%. A downside of denoising is that in the GMM system, it adds nonnegligible processing time.
- 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% in the GMM system. Moreover, the speed for FakeBob to generate an adversarial audio is slowed down 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 only 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 paper is structured as follows. Section II provides the background on a speaker verification system and adversarial attacks (*i.e.*, FakeBob) against speaker verification systems. Section III presents our proposed defense framework and discusses two plugin functions, *i.e.*, denoising and noise-adding. Section IV evaluates the performance of our proposed defense system on normal operations of speaker verification systems and against FakeBob adversarial attacks. Finally, Section V concludes our effort and discusses the future work.

II. BACKGROUND

A. Speaker Verification Systems

Speaker verification (SV) systems have been widely used to identify a person through their voice. In our work, we focus on score-based SV systems, which most state-of-the-art SV systems belong to. A score-based SV system contains two phases: speaker enrollment and speaker recognition, as shown in Figure 1.

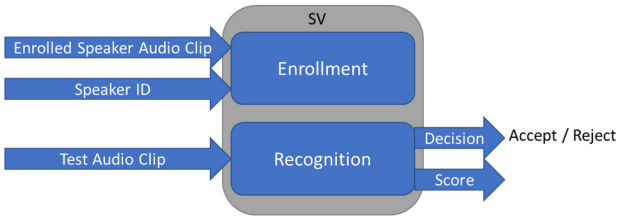


Fig. 1. A Score-Based Speaker Verification System

In the speaker enrollment phase, 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 [13], GMM [14], d-vector [15], and x-vector [16]. In this work, we focus on GMM and i-vector, since they have been widely used in real life and 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 1, 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 similarity 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.

There are two basic false cases for a SV system: (1) accepting a speaker who is not the enrolled user, and (2) rejecting the enrolled speaker. Usually, we call the speaker who 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:

$$FAR = \frac{number\ of\ falsely\ accepted\ illegal\ speaker\ audio\ clips}{total\ number\ of\ all\ illegal\ speaker\ audio\ clips} \quad (1)$$

$$FRR = \frac{number\ of\ falsely\ rejected\ enrolled\ speaker\ audio\ clips}{total\ number\ of\ all\ enrolled\ speaker\ audio\ clips} \quad (2)$$

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. There is tradeoff between them. That is, in general 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 the equal error rate (EER) [17].

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 calculates the similarity scores for all provided audio clips and then finds the threshold that can lead to the EER.

B. Adversarial Attacks on Spekaer Verification Systems

The study of adversarial attacks started from the research of applying deep learning to the image classification problem. In their original work, Szegedy and Goodfellow et al. showed that deep learning is particularly vulnerable to adversarial examples attacks [4] [18]. For example, after added very small perturbations, the image of panda can be recognized as gibbon with 99.3% confidence by a popular deep-learning based classifier. Later, researchers realized that adversarial attacks can be applied to not only deep learning, but also other machine learning methods. Interested readers can refer to the paper [19] for a comprehensive survey on adversarial attacks.

Adversarial attacks and defenses have not been comprehensively and systematically studied in the field of SV systems yet. In the context of a SV system, adversarial attacks attempt to make the SV system falsely accept a well-designed illegal audio, which is called an adversarial audio. Specifically, the adversarial audio is an 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 be the perturbation vector with the same length as x, and x' be the adversarial audio. Then,

$$x' = x + p. \quad (3)$$

When hearing the two audios of x and x', humans may notice no or little difference between them. 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, ε. That is,

$$p = [p_1, p_2, \dots, p_n],\ where\ -\varepsilon < p_i < \varepsilon, 1 \le i \le n \quad (4)$$

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 [5] 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. Specifically, the FakeBob attack is a block-box attack that does not need to know the internal structure of a SV system. Moreover, as shown in [5], FakeBob achieves at least 99% targeted attack success rate (ASR) on both open source and commercial SV systems. ASR can be defined as the following:

$$ASR = \frac{number\ of\ falsely\ accepted\ adversarial\ audios}{total\ number\ of\ all\ adversarial\ audios} \quad (5)$$

Figure 2 shows the basic process of FakeBob adversarial attacks.

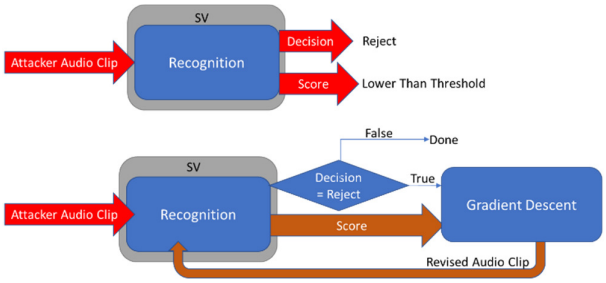


Fig. 2. The Process of FakeBob Adversarial Attacks

Basically, FakeBob applies the basic iterative method (BIM) [20] and the natural evolution strategy (NES) [21] to generate the adversarial audio. That is, the attack takes multiple iterations to get the final adversarial audio (*e.g.*, x') that attempts to minimize the following loss function or objective function

$$L(y) = max\{\theta - S(y), 0\}, \quad (6)$$

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

$$f_G(y) = \nabla_y L(y). \quad (7)$$

Note that here the gradient decent is different from the back-propagation that is 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, *i.e.*,

$$f_{sign}(y_i) = \begin{cases} 1, & if\ y_i > 0 \\ 0, & if\ y_i = 0 \\ -1, & if\ y_i < 0 \end{cases} \quad (8)$$

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^k $(1 \le i \le n)$ is the corresponding signal in the adversarial audio at k^{th} iteration (*i.e.*, x^k), and ε is the perturbation threshold shown in Equation (4). Based on the assumption in Equation (4), a clip function is defined as follows

$$f_c(x_i^k) = \begin{cases} x_i^k, & if\ |x_i^k - x_i| < \varepsilon \\ x_i + \varepsilon, & if\ x_i^k \ge x_i + \varepsilon \\ x_i - \varepsilon, & if\ x_i^k \le x_i - \varepsilon \end{cases} \quad (9)$$

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

$$x^k = f_c(x^{k-1} - lr \times f_{sign}(f_G(x^{k-1}))) \quad (10)$$

where lr is the learning rate. The FakeBob attack is summarized in Algorithm 1.

Algorithm 1 *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* $\boldsymbol{lr}$, *and 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
```

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, applying the FakeBob with the perturbation threshold of 0.002, we obtained an adversarial audio that is falsely accepted by the GMM SV system. Figure 3 shows the time waveform of the adversarial audio and the perturbations (*i.e.*, p in Equation (3)) in both the time domain and the Mel spectrogram. It can be seen that the perturbations used in the FackBob attack are very small (*i.e.*, $-0.002 < p_i < 0.002$) and are similar to white noises (*i.e.*, the perturbations are everywhere with the similar color in the Mel spectrogram). On the other hand, these perturbations are not random, but are intentionally designed to fool the SV system.

III. Proposed Defense System

In this section, we propose a defense system against adversarial attacks in 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 the adversarial audio. Finally, we provide the implementation details 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.

A. Design Goals of a Defense System

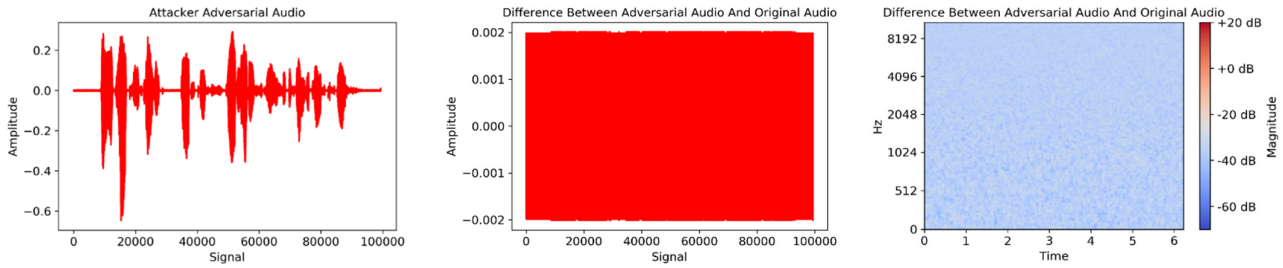
To counteract the adversarial attacks in a SV system, we attempt to design and implement a defense system that achieves the following goals:

- **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 ASR 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 the EER.

B. A Defense System

To achieve these goals, we design a defense system based on the observations from an adversarial audio. Specifically, as shown in Figure 3 in Section II, 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 lose the efficiency against the SV system. Based on this intuition, we propose a defense system as shown in Figure 4. Comparing Figure 4 with Figure 1, it can be seen that we add an additional module, *i.e.*, a 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.

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



(a) Adversarial Audio in Time Domain (b) Perturbations in Time Domain (c) Perturbations in Mel Spectrogram

Fig. 3. An Advesarial Audio in the Time Domain and Its Perturbations in Both the Time Domain and the Mel Spectrogram

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 denoising and noise-adding functions in this work. But other functions can be applied as well, which is our future work.

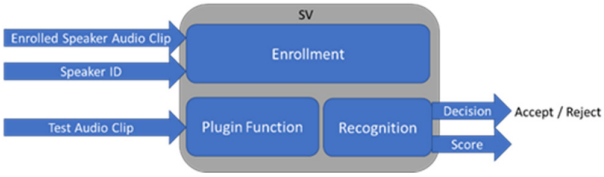


Fig. 4. The Proposed Defense System

C. Denoising

The basic idea of the denoising function is to remove or reduce perturbations or noises in input audios. Here, we applied the method of noise reducing proposed in [8] as our denoising function.

As indicated in [8], 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 the adversarial audio from Figure 3. That is, we consider that the perturbations are Gaussian noises, and p_i in Equation (4) is assumed to have the following normal distribution with a mean of 0 and a standard deviation of σ:

$$p_i \sim N(0, \sigma^2). \tag{11}$$

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. Similarly, inverse STFT (iSTFT) is used to transform the signal from the frequency domain back to the time domain. The algorithm of the denoising function is summarized in Algorithm 2.

Algorithm 2 *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* ***N****, short-time Fourier transform* ***STFT****, inverse short-time Fourier transform* ***iSTFT***

begin
  noise clip ← [0,0, …,0]
  for each signal noise[*i*] *in noise clip*:
    noise[*i*] ← $N(0, \sigma^2)$
  end for
  n_stft ← *STFT*(*noise clip*)
  A_1*_stft* ← *STFT*(A_1)
  A_2*_stft* ← *Remove noise from* A_1 *based on n_stft and* A_1*_stft*
  A_2 ← *iSTFT*(A_2*_stft*)
end

D. Noise-Adding

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

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

$$A_2 = A_1 + Noise \tag{12}$$

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

$$each\ element\ in\ Noise \sim N(0, \sigma^2). \tag{13}$$

The algorithm of the noise-adding function is shown in Algorithm 3.

Algorithm 3 *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 = length of* A_1
  for each signal noise[*i*] *in noise clip*:
    noise[*i*] ← $N(0, \sigma^2)$
  end for
  A_2 ← A_1 + *noise clip*
end

IV. 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 FakeBob attacks.

A. Experimental Setup

The experiments were run parallelly in two virtual machines (VMs) provided by Google Cloud Platform [22] 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 the Kaldi speech recognition toolkit [23] and use the pre-trained models from VoxCelab 1 [24]. Moreover, the adversarial attacks against these SV systems are implemented through FakeBob attacks [5] with the perturbation threshold of 0.002 (*i.e.*, ε = 0.002) and 1,000 maximum iterations (*i.e.*, m = 1,000). The audio dataset used is from LibriSpeech [25] and contains 9 different speakers, which were also applied in FakeBob [5]. For each speaker, there are 100 audio clips, each of which lasts between 4 and 10 seconds. The experiments on SV systems are through the API provided by Kaldi GMM or i-vector.

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 address 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.

B. *Performance Evaluations of the Proposed Defense System for Normal Operations of SV 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 (1) and (2), 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 9 speakers provided, we select two speakers with the identifier of 61 and 2830 as registered users in two separate experiments. As shown in the next section, 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 I 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). In the table, "O" means "Original," "D" means "Denoising," and "A" means "noise-Adding" in the plugin column. 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 increases slightly, especially when σ is not large; for example, $\sigma \leq 0.002$. As a reference, in Table I we also record the threshold of the SV system (*i.e.*, θ). Such a threshold will be applied for studying adversarial attacks. It can be seen that when σ increases, the threshold decreases in general.

It can also be seen from Table I 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 Gaussian 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 II. Similar to results in Table I, 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 I, 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 I. PERFORMANCE EVALUATIONS OF PROPOSED DEFENSE SYSTEM ON NORMAL OPERATIONS OF GMM SV SYSTEMS

Spk	Plugin	σ	EER (%)	Threshold (*i.e.*, θ)	Time (sec)
2830	O	0	1.12	0.0610	17.58
2830	D	0.001	2.25	0.0722	30.91
2830	D	0.002	3.37	0.0708	30.53
2830	D	0.005	2.92	0.0501	31.47
2830	A	0.001	1.35	0.0770	18.77
2830	A	0.002	2.58	0.0454	19.16
2830	A	0.005	5.84	-0.0304	19.50
61	O	0	0.97	0.1285	19.29
61	D	0.001	0.97	0.1215	30.43
61	D	0.002	2.52	0.1369	30.29
61	D	0.005	3.79	0.0911	30.10
61	A	0.001	1.07	0.1098	19.91
61	A	0.002	1.26	0.0810	20.39
61	A	0.005	2.04	0.0270	21.11

TABLE II. PERFORMANCE EVALUATIONS OF PROPOSED DEFENSE SYSTEM ON NORMAL OPERATIONS OF I-VECTOR SV SYSTEMS

Spk	Plugin	σ	EER (%)	Threshold (*i.e.*, θ)	Time (sec)
2830	O	0	0.0	2.1406	390.42
2830	D	0.001	0.0	1.7603	406.68
2830	D	0.002	0.0	1.4734	404.92
2830	D	0.005	0.0	1.5418	404.92
2830	A	0.001	0.0	2.1523	393.14
2830	A	0.002	0.0	1.9582	394.13
2830	A	0.005	0.34	1.5960	394.12
61	O	0	0.0	2.1077	476.48
61	D	0.001	0.29	1.9156	488.05
61	D	0.002	0.1	1.8095	490.72
61	D	0.005	0.97	1.9277	487.47
61	A	0.001	0.87	2.0787	477.55
61	A	0.002	0.78	1.8908	477.64
61	A	0.005	1.94	1.7152	476.89

From both Tables I and II, 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.

C. *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 (5). A smaller value of ASR indicates a better defense performance. Moreover, FakeBob uses multiple iterations to create an adversarial audio, as shown in Algorithm 1. 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 the 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 III 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 σ = 0, 0.001, 0.002, and 0.005. It can be seen that while FakeBob achieves 100% ASR for the original GMM SV system (*i.e.*, σ = 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 σ = 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. It can be clearly seen that noise-adding performs much better than denoising based on the ASR. Moreover, noise-adding leads to a much larger value of the average number of iterations and the 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 σ = 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 attackers' processing speed more than 25 times and makes FakeBob significantly harder to find the adversarial audios.

TABLE III. PERFORMANCE EVALUATIONS OF PROPOSED DEFENSE SYSTEM AGAINST FAKEBOB ATTACKS IN SV SYSTEMS

Spk	Plugin	σ	EER (%)	Avg iter	Avg time (sec)	Avg ASR
GMM						
2830	O	0	1.12	13.5	113.54	100%
2830	D	0.001	2.25	13.4	131.55	79.0%
2830	D	0.002	3.37	12.1	128.85	64.6%
2830	D	0.005	2.92	18.1	186.46	50.7%
2830	A	0.001	1.35	67.0	388.53	23.2%
2830	A	0.002	2.58	634.9	4423.60	6.8%
2830	A	0.005	5.84	701.8	3925.76	6.8%
61	O	0	0.97	32.5	203.82	100%
61	D	0.001	0.97	24.4	252.48	75.4%
61	D	0.002	2.52	33.6	343.07	47.5%
61	D	0.005	3.79	26.5	285.09	51.3%
61	A	0.001	1.07	118.2	841.30	25.5%
61	A	0.002	1.26	575.0	3562.15	3.6%
61	A	0.005	2.04	688.1	4774.93	1.4%
i-Vector						
61	O	0	0.00	76.4	2776.62	100%
61	D	0.002	0.10	57.5	2228.28	40.5%
61	A	0.002	0.78	877.8	31573.95	1.0%

Moreover, we have extended our experiments to an i-vector SV system against FakeBob attacks. As shown in Table III, for speaker 61, denoising and noise-adding with σ = 0.002 are able to reduce the ASR from 100% to 40.5% and 1.0%, respectively. Moreover, the average running time for FakeBob has been significantly increased by applying the noise-adding defense method.

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.*, σ = 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.

V. CONCLUSIONS AND FUTURE WORK

In this work, we have attempted to design and implement a defense system that is simple, light-weight, and effective 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 lose the effect. We have studied two opposite plugin functions, *i.e.*, denoising and noise-adding, and found that noise-adding has much better performance against FakeBob adversarial attacks than denoising. Specifically, noise-adding with σ = 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 σ = 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.*, the noise-adding plugin, should be applied to any SV system to counteract adversarial attacks such as FakeBob. To the best of our knowledge, this is the first attempt in applying the noise-adding method to defend against adversarial attacks in SV systems.

As our on-going work, we will extend our study to other SV systems such as d-vector [15] and x-vector [16]. Moreover, we plan to research the effect of adding other different types of noises, such as Rustle noises [26], on the normal operations of a SV system and against adversarial attacks.

ACKNOWLEDGMENT

This work was supported in part by 2020 Purdue University Fort Wayne Graduate Research Assistantship and 2020 Purdue University Fort Wayne Collaborative Research Grant.

REFERENCES

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

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

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

[21] 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.

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

[23] 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.

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

[25] 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.

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