A New Replay Attack Against Automatic Speaker Verification Systems

SUNG-HYUN YOON, MIN-SUNG KOH, (Member, IEEE), JAE-HAN PARK, AND HA-JIN YU, (Member, IEEE)

1School of Computer Science, University of Seoul, Seoul 02504, South Korea
2School of Computing and Engineering Sciences, Eastern Washington University, Cheney, WA 99004, USA
3Service Laboratory, Institute of Convergence Technology, KT R&D Center, Seoul 06763, South Korea

Corresponding author: Ha-Jin Yu (hjyu@uos.ac.kr)

This work was supported by the 2016 Research Fund of the University of Seoul.

ABSTRACT With the increasing popularity of automatic speaker verification (ASV), the reliability of ASV systems has also gained importance. ASV is vulnerable to various spoofing attacks, especially replay attacks. Thus, recent public competitions and studies based on spoofing attack detection for ASV have mainly focused on the detection of replay attacks. Generally, replayed speech includes the attributes of one playback and two recording devices: the playback device, the recording device used by the attacker, and the recording device embedded in any system to verify input utterances. Therefore, the main attributes differentiating a replayed speech from the genuine speech are the attributes of the playback and the recording devices used by the attacker. In this paper, we propose a novel replay attack and its defense through observation of the general speech-spoofing process. The proposed attack includes only the attribute of one recording device embedded in an ASV system; genuine speech passes through the recording device only once, and the replayed speech produced for the proposed attack passes through the same recording device twice. Because the proposed attack is feasible, it can be considered a new task for replay countermeasures in the training process in order to develop a robust ASV protection system. The experimental results show that this novel replay attack cannot be detected by several of the existing state-of-the-art replay attack detection systems. Furthermore, the new attack can be detected by the same systems successfully if they are retrained with an appropriate dataset designed for the new task.

INDEX TERMS Automatic speaker verification, replay attack, same recording device, spoofing detection.

I. INTRODUCTION

Automatic speaker verification (ASV) is a technique that verifies a user’s identity by analyzing his/her speech. Because it uses only speech, it is relatively convenient, compared to other verification techniques. Recently, it has been widely used in many smart devices that require user verification, such as smart speakers and smartphones. However, ASV is vulnerable to spoofing attacks, such as voice conversion, speech synthesis, and replay attack. Therefore, spoofing detection techniques are required to protect ASV systems from various attacks.

Recently, competitions involving spoofing and ASV countermeasures have been held steadily, and related studies have been conducted [1]–[6]. In the beginning, these challenges and studies mainly focused on detecting spoofing attacks based on logical access, such as voice conversion and speech synthesis. These days, they mainly focus on physical access, such as replay attacks. Spoofing attacks based on logical access have to go through a process of replaying the genuine speech (the target speaker’s original speech) or a spoofed speech (e.g., converted voice or synthesized speech) by using a playback device. In addition, devices that support audio playback and recording functions, such as smartphones, are becoming increasingly popular, allowing individuals to easily record and play back speech samples for replay attacks. Furthermore, replay attacks do not require any special technical knowledge. Accordingly, a replay attack is the easiest to attempt, and can easily deceive ASV systems.

When speech is replayed through a playback device, or recorded on a recording device, its frequency attributes are changed [7]–[12]. Replay attack detection can be regarded...
The conventional replay attack scenario (case A). The attacker surreptitiously records the target speaker’s speech.

as a task that distinguishes the difference in the frequency attributes between genuine and replayed speeches. As shown in Fig. 1, genuine speech entered into an ASV or spoofing detection system has already passed through the recording device embedded in the system (rec_sys) once. The type of rec_sys may be the same, regardless of the user (e.g., all users have the same smart speakers) or may be different, depending on the users’ recording device (e.g., they use their own smartphones). In both cases, genuine speech has the attribute of rec_sys. In contrast, replayed speech (denoted by “Spoof” in Fig. 1) has the additional attributes of the playback device (play_atk) and recording device (rec_atk) used by the attacker, although they may differ slightly, depending on the replay attack methods. Based on these facts, most studies on replay attack detection focus on how to better capture differences in frequency attributes between the genuine and replayed speeches with play_atk and rec_atk.

As described above, the replay attack detection task has been focusing on distinguishing the differences generated by play_atk and rec_atk. Therefore, the attacker wants to make sure the attributes generated by play_atk and rec_atk are not included in the replayed speech. A way to ensure that the attributes of rec_atk are not included in the replayed speech sample is to record using the same recording device as rec_sys, or to steal a digital copy of the genuine speech. One way to ensure that the attributes of play_atk are not included in replayed speech is to play it back using a mouth simulator, not play_atk. A mouth simulator is a hardware device that can closely replicate the human voice. It should be designed to satisfy the standards related to an artificial mouth (e.g., ITU-T Rec. P.51 as mentioned in [13]). Also, it is used where speech close to human speech is required. The speech generated by a mouth simulator plays a role in reference speech. For example, many companies use the mouth simulator to test their speech recognizers [14]. The attributes of human speech and those of speech generated by a mouth simulator are almost the same [15]–[17]. Therefore, replayed speech produced in this way includes only the attributes of rec_sys, like genuine speech. It can be difficult in some cases to steal a digital copy of the genuine speech because hacking expertise may be required. However, it is not difficult to record speech using the same recording device as rec_sys, and to play it back using a mouth simulator. In most cases, rec_sys corresponds to the recording device embedded in a smart device, but there are only a few dominant smart devices on the market. Each dominant smart device is mass-produced using the same production/manufacturing process, and they are sold in various markets around the globe. Therefore, the recording device could be same as rec_sys and mouth simulators can be purchased in those markets without difficulty.

The conventional replay attack detection systems cannot detect the new replay attack in this paper, because they are trained to detect a conventional replay attack in which the replayed speech includes the attributes of play_atk and/or rec_atk. The proposed new replay attack, which has only the attributes of rec_sys, can effectively deceive the conventional replay attack detection systems without difficulty. Therefore, we have to be prepared to detect this new replay attack. The difference between genuine and new replayed speeches is the number of times a speech passes through rec_sys. To distinguish the number of times a speech passes through rec_sys, the features of the speech should differ, depending on the number of times. We propose this task as a new replay attack countermeasure, and evaluate the performance of the proposed task by using state-of-the-art spoofing detection systems. The experimental results show that although the attributes included in the replayed speech differ depending on the number of passes through the recording device, the detection systems for conventional replay attacks cannot distinguish this difference properly. However, we also show that the systems that failed to detect the proposed replay attack can become a successful countermeasure after retraining with an appropriate dataset for the new task. This means that a dataset appropriate for the new task is required in order to build a system to detect this new attack.

The remainder of this paper is organized as follows. Section II describes the conventional replay attacks. Section III discusses the vulnerability in the conventional task, and proposes a new task for replay attack detection. Section IV describes the experimental setup and configurations of the dataset used. Section V analyzes the obtained results. Finally, Section VI concludes the paper.

II. CONVENTIONAL REPLAY ATTACKS

As mentioned earlier, the attribute of rec_sys is included in both the genuine and the replayed speeches. The replayed speech further includes the attributes of the devices used during the replay attack, which are slightly different depending on the method of the attempted replay attack. There are two major methods for executing replay attacks in practice.

A. REPLAY ATTACK: CASE A

After the replayed speech is entered into the system, it includes the attributes of rec_atk, play_atk, and rec_sys. The rec_atk may be a different device than rec_sys. The replay attack process is as follows (Fig. 1).

1) An attacker surreptitiously records the target speaker’s speech using any type of recording device, denoted as
At this time, the attribute of \( \text{rec}_\text{atk} \) is added to the recorded speech.

2) The attacker uses any type of playback device to play back the stolen speech sample, and the replayed speech has the attribute of \( \text{play}_\text{atk} \).

3) After the replayed speech enters the system, the attribute of \( \text{rec}_\text{sys} \) is added to the replayed speech sample entering the system.

This process corresponds to the physical access scenario described in [6], where the replayed speech passes through \( \text{rec}_\text{atk}, \text{play}_\text{atk} \), and \( \text{rec}_\text{sys} \) only once.

**B. REPLAY ATTACK: CASE B**

As shown in Fig. 2, the replayed speech sample entered into the system includes the attributes of \( \text{play}_\text{atk} \) and \( \text{rec}_\text{sys} \), but not \( \text{rec}_\text{atk} \). In this scenario, several conditions are required.

The replay attack process is as follows (Fig. 2).

1) An attacker steals a digital copy of genuine speech. This corresponds to the stolen voice scenario, where the attacker does not need to use any recording device [2]–[5]. The stolen speech sample includes only the attribute of \( \text{rec}_\text{sys} \) because it is exactly the same as the genuine speech entered into the system.

2) Similar to case 1, the attacker uses any device to play back the stolen speech, which adds the attribute of \( \text{play}_\text{atk} \). The replayed speech sample now includes the attributes of \( \text{rec}_\text{sys} \) and \( \text{play}_\text{atk} \).

3) The replayed speech sample enters the system. However, because it already has the attribute of \( \text{rec}_\text{sys} \), it includes the attribute of \( \text{rec}_\text{sys} \) twice, along with that of \( \text{play}_\text{atk} \).

This process requires more effort by the attacker, but it deceives the system much more effectively than the method in Section II-A. In this case, note that the replayed speech passes through \( \text{play}_\text{atk} \) once and \( \text{rec}_\text{sys} \) twice.

The differences in the attributes of the recording device between the genuine and replayed speeches are easily minimized. Note that even if an attacker does not have the expertise to access the system to steal a digital copy of the genuine speech, he/she can just use \( \text{rec}_\text{atk} \) with the same \( \text{rec}_\text{sys} \). Thus, the major difference between genuine and replayed speeches is the attribute of \( \text{play}_\text{atk} \).

III. NEW TASK FOR REPLAY ATTACK DETECTION

Consider a case in which the replayed speech does not include the attribute of \( \text{play}_\text{atk} \) but only that of \( \text{rec}_\text{sys} \). It can be sufficiently feasible as follows (see Fig. 3 and Algorithm 1).

1) As explained in Section II-B, an attacker steals a digital copy of the genuine speech, or surreptitiously records the target speaker’s speech using the same recording device as \( \text{rec}_\text{sys} \). Only the attribute of \( \text{rec}_\text{sys} \) is included in the stolen speech.

2) To play speech that sounds genuine, the attacker uses a mouth simulator, which is a hardware device that can produce sound very close to the human voice. The replayed speech sample using the mouth simulator does not include the attribute of \( \text{play}_\text{atk} \).

3) The replayed speech passes through \( \text{rec}_\text{sys} \) while entering the system. Because \( \text{rec}_\text{sys} \) is the device that the speech has already passed once in Step 1, the replayed speech includes only the attribute of \( \text{rec}_\text{sys} \), even after entering the system.

This new replay attack means an attacker can exclude any attribute of \( \text{play}_\text{atk} \) from replayed speech. Despite this new attack being feasible in practical situations, to the best of our knowledge, defending against it has not been addressed yet.

To distinguish this new replayed speech from conventional replayed speech (described in Section II), we denote this new replayed speech with \( \text{rec2pass} \) speech in this paper. Note that \( \text{rec2pass} \) speech has only the attribute of \( \text{rec}_\text{sys} \), like a genuine speech sample. The only difference between genuine and \( \text{rec2pass} \) speeches is that genuine speech passes through \( \text{rec}_\text{sys} \) only once, whereas \( \text{rec2pass} \) speech passes through it twice. For a replay attack detection system to distinguish between genuine and \( \text{rec2pass} \) speeches, the features of the speech have to be different, depending on the number of times the speech passes through \( \text{rec}_\text{sys} \).

We use the convolutional restricted Boltzmann machine (ConvRBM) [18]–[20] to visualize the difference in the number of passes through the recording device. The filters (weights) of the trained ConvRBM can represent sub-band filters, which resemble auditory gammatone filters [21].
Algorithm 1 The New Replay Attack Scenario

| Step | Attacker | Target speaker |
|------|----------|----------------|
| 1.   | starts speaking to system | stops speaking. |
| 2.   | a) surreptitiously records the genuine speech using the same recording device embedded in rec_sys. | |
| 3.   | b) steals a digital copy of the genuine speech. | |
| 4.   | plays the genuine sample for the system using a mouth simulator | |
| 5.   | attacker only needs to use one method, a) or b. | |

IV. EXPERIMENTS

We conducted two experiments using the existing methods. First, we checked whether the replay attack detection systems for the conventional tasks also work well for the new task. Second, we rebuilt and evaluated replay attack detection systems for the new task by using well-known state-of-the-art methods.

A. DATABASE

We used a text-dependent dataset with the Korean phrase “nae mogsoriro injeung”\(^1\) for genuine speech, which was collected using the KT GiGA Genie smart speaker. This dataset comprises 16,191 clean speech samples from

---

\(^1\)It means “verify by my voice” in English.
The average duration of all speech samples was about 2.3 s. The distance between the speaker and the recording device (i.e., Giga Genie) was approximately 1 m. We used 1,431 genuine speech samples for training and the remaining 14,760 for evaluation. There was no speaker overlap across the training and evaluation datasets; the training set comprised 18 male and 18 female speakers, and the evaluation set comprised 185 male and 184 female speakers.

The replayed speech dataset was collected by replaying the genuine dataset using three smartphones: the Apple iPhone 6S, the LG V30, and the Samsung Galaxy S6. Samples were replayed with no background noise, and the distance between the smartphone and the recording device was approximately 1 m. The number of replayed speech samples used for training was 1,350, and all of them were replayed by the Galaxy S6. The number of replayed speech samples used for evaluation was 28,518, which were replayed by the iPhone 6S (14,389) or the LG V30 (14,129). The speaker partitioning of the replayed dataset was the same as that of the genuine dataset.

The rec2pass speech dataset was collected by replaying the genuine dataset with a mouth simulator. The mouth simulator we used is the B&K Mouth Simulator Type 4222 [13]. These samples were also replayed without background noise, and the distance between the mouth simulator and the recording device was about 1 m. The rec2pass training set comprised 1,431 speech samples collected from 18 male and 18 female speakers. The speaker partitioning of the rec2pass training dataset was the same as that of training dataset of genuine samples. The rec2pass evaluation set comprised 1,576 speeches collected from 8 male and 32 female speakers, and the 40 speakers were a subset of the speakers of the genuine evaluation set.

All speech samples were recorded in wav format at a 16-kHz sampling rate with a 16-bit resolution per sample. The average duration of all speech samples was about 2.3 s (varying between 1.63 s and 3.66 s), including silences of about 0.5 s at both ends. Tables 1 and 2 show the number of speakers and utterances of datasets respectively, in the datasets.

**B. EXPERIMENTAL SETUP**

We used five state-of-the-art methods to evaluate the conventional replay attack detection task. Regardless of the feature extraction method, we did not perform voice activity detection on all speech samples, because silent segments are also useful for spoofing detection [22]. Mean and variance normalization for features were also not applied, if not mentioned otherwise. For each method, we trained two types of system. One was trained using the genuine and replayed dataset (sys_con), which was evaluated for conventional tasks and the new task. The other was trained using the genuine and rec2pass datasets (sys_new) and was evaluated for the new task only. Unless otherwise noted, the training conditions were the same regardless of the system type.

1) ConvRBM-CC + GMM

Convolutional restricted Boltzmann machine cepstral coefficients (ConvRBM-CCs) [19] are similar to Mel frequency cepstral coefficients (MFCCs), except that ConvRBM-CCs are extracted using the filterbank obtained from a trained ConvRBM instead of a Mel filterbank. Details of the extraction process for ConvRBM-CCs are shown in [19].

We performed pre-processing for each speech sample, removing the DC offset (subtracting the global mean in the time domain) followed by pre-emphasis filtering with coefficient 0.97. Each speech sample used to train the ConvRBM was normalized to zero mean and unit variance. The number of filters was 60, and the length of each filter was 128. The filters were initialized from a normal distribution with a standard deviation of 0.001. A noisy leaky rectified linear unit (NLReLU) function with parameter 0.01 was used for sampling hidden units [23]. The Adam optimizer [24] with a learning rate of 0.0005 was used for training. The batch size was set as 16. The training epochs chosen were 15 and 17 for sys_con and sys_new, respectively. Tensorflow [25] was used to implement the network.

The dimensionality of the feature vector was 90: 30 ConvRBM-CC + delta + acceleration. Average pooling was applied with a 25 ms window and a 10 ms shift.

We trained a Gaussian mixture model (GMM) using the features from both genuine and spoofed (replayed or rec2pass) speech samples, which comprised 512 mixture components with diagonal covariance. The model was trained for 10 iterations of the expectation-maximization (EM) algorithm; then, it was used to adapt the genuine and spoofed GMMs by using maximum a posteriori (MAP) [26]. Scores were computed as the log likelihood ratio. The Kaldi speech recognition toolkit [27] was used to implement the GMMs.

2) CQCC + GMM

The constant Q cepstrum coefficient (CQCC) [28], [29] is a widely used feature in the field of spoofing detection, capturing more detailed information for spoofing detection than conventional features (e.g., MFCCs). It is extracted using the constant-Q transform (CQT) [30] instead of the Fourier

---

**TABLE 1. Number of speakers in the datasets.**

|         | Training | Evaluation |
|---------|----------|------------|
| Male    | Female   | Male       | Female   |
| Genuine |          | 185        | 184      |
| Rec2pass|          | 18         | 18       |
| Replayed|          | 185        | 184      |

**TABLE 2. Number of utterances in the datasets.**

|         | Training | Evaluation |
|---------|----------|------------|
| Male    | Female   | Male       | Female   |
| Genuine |          | 711        | 720      |
| Rec2pass|          | 716        | 717      |
| Replayed|          | 662        | 688      |

(7276 / 7276) (V30 / Phone6) (6853 / 7113) (V30 / Phone6)
transform when converting a speech signal to the frequency domain.

We extracted 29-dimensional CQCCs. The CQT was applied with a maximum frequency of 8 kHz (Nyquist frequency), a minimum frequency of 7.8125 Hz (=8,000/210) and 96 bins per octave. The resampling period was 16. We first appended C0, and then delta and acceleration to extract 90-dimensional features. The Matlab toolbox [31] was used to extract CQCCs.

Similar to method 1 (ConvRBM + GMM), we used GMMs as a binary (genuine or spoofed) classifier. All the training conditions of the GMMs and the scoring method were the same as those described in Method 1.

3) DNN-FBCC + GMM
The deep neural network filterbank cepstral coefficient (DNN-FBCC) was proposed in [32] to capture the differences between genuine and synthetic speeches more effectively. It is similar to the MFCC, except that the DNN-FBCC is extracted using the filterbank trained by a filterbank neural network (FBNN) [32]. A FBNN consists of two fully connected hidden layers, followed by a fully connected Softmax classifier. It is trained to classify speech as genuine or spoofed at the frame level. The weight matrix of the first hidden layer of the FBNN is restricted by a non-negative function, and is band-limited using a mask matrix. It is used as a filterbank matrix to extract the DNN-FBCC.

The power spectrogram (before applying the logarithm) was extracted from a pre-processed speech signal using a Hamming window 25 ms in length with a 10 ms shift. The pre-processing method was the same as Method 1. The number of FFT points was 1,024.

The FBNN was trained by using the frames in training speeches. All weights of the FBNN were initialized from a LeCun normal distribution [33]. In the first and second hidden layers, there were 128 and 100 units, respectively. The sigmoid function was applied as a non-negative function, and a linear triangle filterbank matrix was used as the band-limiting mask matrix for the weights of the first hidden layer. Sigmoid activation was applied for the second hidden layer. We used an Adam optimizer with a learning rate of 0.001. The training epochs chosen were 97 and 56 for sys_new and sys_con, respectively. Tensorflow was used to implement the network.

We extracted 40-dimensional features, comprising delta and acceleration of 20-dimensional static DNN-FBCCs, so the static features were not used.

A GMM-based binary classifier was also used. All training conditions for the GMMs and the scoring method were the same as those described in methods 1 and 2.

4) GD-GRAM + RESNET-18
In [34], the group delay gram (GD-gram) was used for spoofing detection. The GD-gram involves concatenating the group delay of all frames, such as a spectrogram that involves concatenating the spectra of all frames. The group delay is defined as the negative gradient of an unwrapped phase spectrum.

As in Method 1, pre-processing was applied to each speech sample. We extracted 1,025-dimensional GD-grams using 2,048 FFT points. The GD-grams were truncated or padded along the time axis only, so each GD-gram had a dimension of 256 × 1025.

We used ResNet-18 [35] to model the GD-gram and a fully connected layer with Softmax as a binary classifier. Unlike in [34] which used a pre-trained network, all weights of ResNet-18 and the classifier were initialized from a LeCun normal distribution. The attention mechanism was not applied. The learning rate was fixed at 0.000001, and no dropout was applied. The training epochs were 8 and 42 for sys_con and sys_new, respectively. Tensorflow was used to implement the network.

V. RESULTS
We used equal error rate (EER) as an evaluation metric for all experiments. In order to ensure the reliability of our results, we first evaluated the performance of our systems under trials of ASVspoof 2017 version 2 and ASVspoof 2019 PA, which have been widely used in the studies on replay attack detection. Notice that both datasets contain the replayed speeches which are produced in conventional ways described in Section II-A (corresponding to ASVspoof 2019 PA) and Section II-B (corresponding to ASVspoof 2017). However, the two datasets do not have rec2pass speeches produced by Algorithm 1 in Section III. In other words, both datasets are for the task of conventional replay attack detection (i.e., distinguishing between genuine and conventionally replayed speeches), not for our proposed task (i.e., distinguishing between genuine and rec2pass speeches). Furthermore, it must be stressed that the ASVspoof 2017 and ASVspoof 2019 PA datasets providing conventionally replayed speeches cannot be tested for the
proposed new task (i.e., distinguishing rec2pass speech from genuine speech) because the same recording systems of the two datasets to make rec_sys are not available in public (e.g., recording environments and devices etc.). Therefore, the ASVspoof 2017 and ASVspoof 2019 PA datasets were evaluated only for the conventional replay attacks.

Table 3 shows the performance of the aforementioned five replay detection systems for the evaluation trials of ASVspoof 2017 version 2 and ASVspoof 2019 PA. The experimental setup used to obtain the data in Table 3 may have differed slightly from those described in Section IV-B. For example, cepstral mean and variance normalization (CMVN) was applied for the ASVspoof 2017 dataset, but not for the ASVspoof 2019 PA dataset. This was in accordance with the experimental setups of the baseline presented by each organizer. Note that the performance of system 2 (CQCC + GMM) is the baseline performance introduced by the organizers of the ASVspoof 2017 and 2019 competitions.

In the trials of the ASVspoof 2017, systems 1 and 3 showed similar EERs to system 2, but systems 4 and 5 showed higher EERs than system 2. In particular, system 4 showed an EER quite a bit higher than the other systems. In [34], system 4 showed a perfect EER (i.e., 0%) in trials of ASVspoof 2017 version 1. However, that system did not result in a perfect EER in our experiment. One of the main differences between our experiment and [34] for system 4 is that we did not apply an attention mechanism to system 4. The attention mechanism applied in [34] should be supported by high classification accuracy. In our experiments, however, the classification accuracy (related to EER) of the system 4 was not high enough, especially for the trials of ASVspoof 2017 version 2. Low accuracy results in the wrong attention mechanism, because the attention map to apply depends on the classification results of the model. For example, the attention map for genuine speech can be applied to replayed speech, or vice versa. The different is that we did not use a pre-trained model as in [35]. For a fair comparison with the other systems that were trained using only ASVspoof 2017 or 2019 PA datasets, we trained a random initialized model rather than the pre-trained model used in [34]. Regarding the trials of ASVspoof 2019 PA, systems 1 and 3 showed quite a bit higher EERs than system 2. System 4 showed the lowest EER, compared to the other systems. System 5 showed a slightly higher EER than system 2.

Compared to the baseline system (i.e., system 2), we can see that systems 1 and 3 cannot properly capture the attributes of the recording device included in replayed speech, but can capture the attribute of the playback device as much as the baseline system. Poor results with systems 4 and 5 for ASVspoof 2017 version 2 in Table 3 imply that systems 4 and 5 cannot properly detect replay attacks having no attributes of recording devices. In contrast, the good result from system 4 for ASVspoof 2019 PA in Table 3 means that system 4 is better at detecting replay attacks having attributes of both playback and recording devices.

Table 4 shows the performance of the systems trained using the genuine and replayed datasets which were collected using the AI speaker (KT Giga Genie) described in Section IV-A, where the replayed dataset is reproduced by using the conventional replay attack method (sys_con). In other words, these systems were trained to detect conventional replay attacks. Note that the EERs were computed on the conventional replay attack detection task (genuine/replayed), on the other hand, the error rates in the third column were computed on the proposed task (genuine/rec2pass). We computed the error rates using the same thresholds used to compute the EERs.

In Table 4, systems 1, 2, and 5 showed substantially low EERs on the conventional task. However, systems 3 and 4 showed a lot higher EERs than the other systems, which means that systems 3 and 4 are not reliable as a replay attack detector, at least on the task with the data we collected. System 3 showed especially poor performance on both the conventional and the proposed replay attack detection tasks. The column headed “Error rate on the proposed task” indicates how well the systems detect rec2pass attacks. Hence, the high error rates imply that the corresponding systems failed to detect the new replay attack proposed in this paper. Elaborating on Table 4, all systems except system 4, could not detect the rec2pass attacks at all, which means that most systems for conventional replay attack detection failed to detect the new rec2pass attacks. Although only system 4 showed a relatively low error rate for rec2pass attack detection, it is not reasonable to assume that the generalization performance of system 4 is good, because its EER was much higher than those of the other systems (i.e., system 1, 2, and 5). Rather, it seems that system 4 does not effectively detect either conventional replay or rec2pass attacks. In summary, the results in Table 4 show that many state-of-the-art systems are very much vulnerable to the new replay attack, even well-trained systems for conventional replay attack.

Table 5 shows the performance of the same systems after training them with the genuine and rec2pass datasets.

---

**TABLE 3.** EERs of the systems for the evaluation trials of the ASVspoof 2017 ver.2 and ASVspoof 2019 PA.

| System | ASVspoof 2017 ver. 2 | ASVspoof 2019 PA |
|--------|---------------------|-----------------|
| 1 (ConvRBM-CC + GMM) | 14.8629 | 16.5522 |
| 2 (CQCC + GMM) | 13.74 | 9.87 |
| 3 (DNN-FBCC + GMM) | 15.4709 | 19.9774 |
| 4 (GD-gram + ResNet-18) | 35.3526 | 6.1261 |
| 5 (Spectrogram + LSTM) | 21.0602 | 11.926 |

**TABLE 4.** EERs of systems trained to detect conventional replay attacks, and error rates on the proposed task. The errors were measured using the threshold obtained for the EERs.

| System | EER on the conventional task (%) | Error rate on the proposed task (%) |
|--------|---------------------------------|-----------------------------------|
| 1 (ConvRBM-CC + GMM) | 1.7471 | 100 |
| 2 (CQCC + GMM) | 0.6672 | 100 |
| 3 (DNN-FBCC + GMM) | 12.4522 | 100 |
| 4 (GD-gram + ResNet-18) | 11.2805 | 24.6827 |
| 5 (Spectrogram + LSTM) | 0.5841 | 100 |
In contrast, both sys_con and rec2pass was trained using genuine, replayed, and proposed replay attacks. Because we trained them using the dataset appropriate for the proposed task, unlike the results in Table 4, all systems showed significantly low EERs on the proposed task. This means that the state-of-the-art replay attack detection systems in Table 4, which were all vulnerable to the proposed new replay attack, are now well-prepared against the new rec2pass attacks after being trained for the proposed new task. However, as shown in the 3rd column of Table 5, all systems failed to detect the conventional replay attacks because they were trained without the dataset of the conventional replay attacks. EER results of the same systems trained with both the conventional replay attack and the new replay attack datasets are shown in Table 6.

Table 6 shows the performance of the same systems on the task of distinguishing between genuine and spoofed speech, where spoofed speech includes both conventional replay attacks and the rec2pass attack. In type 1, each system was trained using genuine, replayed, and rec2pass datasets. In contrast, both sys_con (i.e., in Table 4) and sys_new (i.e., in Table 5) were used in types 2 and 3. Type 2 corresponds to score-level fusion, in which the final score is computed by weighted sum of the scores from sys_con (denote as $s_{\text{con}}$) and sys_new (denote as $s_{\text{new}}$). The weights were estimated by linear regression using the training dataset. Type 3 is the result of decision-level fusion: $\theta_{\text{con}}$ and $\theta_{\text{new}}$ denote the thresholds for sys_con and sys_new, respectively. Suppose that both $s_{\text{con}} > \theta_{\text{con}}$ and $s_{\text{new}} > \theta_{\text{new}}$ are satisfied (i.e., classified as genuine by both systems), then the final score is $s_{\text{con}} + s_{\text{new}}$. All other cases of type 3 were scored as follows: if $s_{\text{con}} \leq \theta_{\text{con}}$ (i.e., classified as replayed by sys_con), then the final score is $s_{\text{con}}$. Otherwise, the final score is $s_{\text{new}}$ (i.e., classified as rec2pass by sys_new). By rebuilding the systems with the replayed and rec2pass datasets together as a spoofed dataset, we confirmed that all systems can distinguish between genuine and spoofed speech. In particular, type 3 showed the lowest EERs for all systems except system 1. To sum up, the dataset appropriate for the proposed new task is also required in order to build a robust countermeasure against both the conventional replay attacks and the feasible, new replay attack presented in this paper.

### VI. CONCLUSION

A new task for replay countermeasures has been proposed in order to be prepared against a new feasible replay attack presented in this paper. Under the new task, the difference between genuine and replayed speeches has been investigated and observed: genuine speech passes the recording device that is embedded in a system only once, whereas replayed speech passes the same recording device twice. Even though this condition allows an attacker to deceive a system much more effectively, this new replay attack has not yet been addressed in previous ASV spoofing challenges and studies on replay attack detection.

The new replay attack could not be detected by the existing replay countermeasures trained for conventional replay attacks. To tackle the new replay attack, we have confirmed that the characteristics of the new replayed speeches differ from those of genuine speeches, depending on the number of times the sample passes through recording devices. This observation indicates that the new replay attack, which fools many state-of-the-art systems, can be detected successfully by retraining the same system with an appropriate dataset built for the proposed new task. To improve the reliability of many conventional replay attack detection systems, the observed new attack must be considered in the training process so that the system is well prepared for this new, feasible replay attack.

### REFERENCES

[1] Z. Wu, T. Kinnunen, N. Evans, J. Yamagishi, C. Hanilci, M. Sahidullah, and A. Sizov, “ASVspoof 2015: The first automatic speaker verification spoofing and countermeasures challenge,” in Proc. Interspeech, 2015, pp. 1–5.

[2] P. Korshunov, S. Marcel, H. Muckenhurn, A. R. Goncalves, A. G. S. Mello, R. P. V. Violato, F. O. Simoes, M. U. Neto, M. de Assis Angeloni, J. A. Stuchi, H. Dinkel, N. Chen, Y. Qian, D. Paul, G. Saha, and M. Sahidullah, “Overview of BTAS 2016 speaker anti-spoofing competition,” in Proc. IEEE 8th Int. Conf. Biometrics Theory, Appl. Syst. (BTAS), Sep. 2016, pp. 1–6.

[3] T. Kinnunen, M. Sahidullah, H. Delgado, M. Todisco, N. Evans, J. Yamagishi, and K. A. Lee, “The ASVspoof 2017 challenge: Assessing the limits of replay spoofing attack detection,” in Proc. Interspeech, Aug. 2017, pp. 1–6.

[4] H. Delgado, M. Todisco, M. Sahidullah, N. Evans, T. Kinnunen, K. A. Lee, and J. Yamagishi, “ASVspoof 2017 version 2.0: Meta-data analysis and baseline enhancements,” in Proc. Odyssey Speaker Lang. Recognit. Workshop, Jun. 2018, pp. 1–9.

[5] T. Kinnunen, M. Sahidullah, M. Falcone, L. Costantini, R. G. Hautamaki, D. Thomsen, A. Sarkar, Z.-H. Tan, H. Delgado, M. Todisco, N. Evans, V. Hautamaki, and K. A. Lee, “RedDots replayed: A new replay spoofing attack corpus for text-dependent speaker verification research,” in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Mar. 2017, pp. 5395–5399.

[6] (2019). ASVspoof: Automatic Speaker Verification Spoofing and Countermeasures Challenge Evaluation Plan. [Online]. Available: http://www.asvspoof.org/asvspoof2019/asvspoof2019_evaluation_plan.pdf
[7] M. S. Saranya and H. Murthy, “Decision-level feature switching as a paradigm for replay attack detection,” in Proc. Interspeech, Aug. 2018, pp. 686–690.

[8] K. Srisankardara, V. Sethu, and E. Ambikairajah, “Deep siamese architecture based real-time replay detection for secure voice biometric,” in Proc. Interspeech, Aug. 2018, pp. 671–675.

[9] T. Gunendrudasan, B. Wickramasinghe, N. P. Le, E. Ambikairajah, and J. Epps, “Detection of replay-spoofing attacks using frequency modulation features,” in Proc. Interspeech, Aug. 2018, pp. 636–640.

[10] M. Kamble, H. Tak, and H. Patil, “Effectiveness of speech demodulation-based features for replay detection,” in Proc. Interspeech, Aug. 2018, pp. 641–645.

[11] G. Suthokumar, V. Sethu, C. Wijenayake, and E. Ambikairajah, “Modulation dynamic features for the detection of replay attacks,” in Proc. Interspeech, Aug. 2018, pp. 691–695.

[12] M. Kamble and H. Patil, “Novel variable length energy separation algorithm using instantaneous amplitude features for replay detection,” in Proc. Interspeech, Aug. 2018, pp. 646–650.

[13] Bruel & Kjaer, Nærum, Denmark. Product Data: Mouth Simulator Types 4227 and 4227-A. Accessed: Aug. 21, 2019. [Online]. Available: https://www.bksv.com//media/literature/Product-Data/bp0505.ashx.

[14] J. Cohen, W. Barton, J. Ploumis, and D. J. Ely, “Automated testing of voice recognition software,” U.S. Patent 7 567 019, Jul. 14, 2009.

[15] Specification and Measurement of Speech Transmission Quality: Part 2: Mouth-to-Ear Speech Transmission Quality Including Terminals, Standard ETSI, EG. 201 377-2, 2004.

[16] Head and Torso Simulator for Telephony, document ITU-T P. 58, 2013.

[17] Use of Head and Torso Simulator (HATS) for Hands-Free and Handset Terminal Testing, document ITU-T P. 58, 2009.

[18] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representation,” in Proc. 26th Annua. Int. Conf. Mach. Learn., 2009, pp. 669–676.

[19] H. B. Sailor and H. A. Patil, “Novel unsupervised auditory filterbank learning using convolutional RBM for speech recognition,” IEEE/ACM Trans. Audio, Speech, Language Process., vol. 24, no. 12, pp. 2341–2353, Dec. 2016.

[20] H. Sailor, M. Kamble, and H. Patil, “Auditory filterbank learning for temporal modulation features in replay spoof speech detection,” in Proc. Interspeech, Aug. 2018, pp. 666–670.

[21] R. Patterson, I. Nimmo-Smith, J. Holdsworth, and P. Rice, “An efficient auditory filterbank based on the gammatone function,” in Proc. Meeting Group on Audio and Speech Processing, no. 7, pp. 1–5.

[22] M. Sahidullah, T. Kinnunen, and C. Hanilci, “A comparison of features for synthetic speech detection,” in Proc. Interspeech, 2015, pp. 1–6.

[23] A. L. Mass, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in Proc. ICML, 2013, vol. 30, no. 1, pp. 1–6.

[24] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, arXiv:1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980.

[25] M. Abadi et al., “TensorFlow: A system for large-scale machine learning,” in Proc. 12th USENIX Symp. Oper. Syst. Design Implement. (OSDI), 2016, pp. 265–283.

[26] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, “Speaker verification using adapted Gaussian mixture models,” Digit. Signal Process., vol. 10, nos. 4–6, pp. 19–41, Jan. 2000.

[27] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Siuokvy, G. Stemmer, and K. Vesely, “The Kaldi speech recognition toolkit,” in Proc. Automat. Speech Recognit. Understand. (ASRU) Workshop, 2011, pp. 1–4.

[28] M. Todesico, H. Delgado, and N. Evans, “A new feature for automatic speaker verification anti-spoofing: Constant Q cepstral coefficients,” in Proc. Odyssey, vol. 45, Jul. 2016, pp. 283–290.

[29] M. Todesico, H. Delgado, and N. Evans, “Constant-Q cepstral coefficients: A spoofing countermeasure for automatic speaker verification,” Comput. Speech Lang., vol. 9, pp. 516–535, Sep. 2017.

[30] J. C. Brown, “Calculation of a constant Q spectral transform,” J. Acoust. Soc. Amer., vol. 89, no. 1, pp. 425–434, Jan. 1991.

[31] C. Schorkhuber, A. Klupuri, N. Holighaus, and M. Dorfler, “A MATLAB toolbox for efficient perfect reconstruction time-frequency transforms with log-frequency resolution,” in Proc. 53rd Int. Conf. Semantic Audio, Audio Eng. Soc., 2014, pp. 1–8.

[32] H. Yu, Z.-H. Tan, Y. Zhang, Z. Ma, and J. Guo, “DNN filter bank cepstral coefficients for spoofing detection,” IEEE Access, vol. 5, pp. 4779–4787, 2017.

[33] G. Klamber, U. Unterrhiner, A. Mayr, and S. Hochreiter, “Self-normalizing neural network,” in Proc. Adv. Neural Inf. Process. Syst. (NIPS), 2017, pp. 979–989.

[34] F. Tom, M. Jain, and P. Dey, “End-to-end audio replay attack detection using deep convolutional networks with attention,” in Proc. Interspeech, Aug. 2018, pp. 681–685.

[35] K. H. Zheng, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.

[36] S. Hochreiter and J. Schmidhuber, “Long short term memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.

[37] M. Henaff, A. Szlam, and Y. LeCun, “Recurrent orthogonal networks and long-memory tasks,” 2016, arXiv:1602.06662. [Online]. Available: https://arxiv.org/abs/1602.06662.

[38] R. Jozefowicz, W. Zaremba, and I. Sutskever, “An empirical exploration of recurrent neural network architectures,” in Proc. Int. Conf. Mach. Learn. (ICML), 2015, pp. 2342–2350.