Vocoder drift in x-vector–based speaker anonymization

Michele Panariello, Massimiliano Todisco, Nicholas Evans

EURECOM, Sophia Antipolis, France
firstname.lastname@eurecom.fr

Abstract

State-of-the-art approaches to speaker anonymization typically employ some form of perturbation function to conceal speaker information contained within an x-vector embedding, then resynthesize utterances in the voice of a new pseudo-speaker using a vocoder. Strategies to improve the x-vector anonymization function have attracted considerable research effort, whereas vocoder impacts are generally neglected. In this paper, we show that the impact of the vocoder is substantial and sometimes dominant. The vocoder drift, namely the difference between the x-vector vocoder input and that which can be extracted subsequently from the output, is learnable and can hence be reversed by an attacker; anonymization can be undone and the level of privacy protection provided by such approaches might be weaker than previously thought. The findings call into question the focus upon x-vector anonymization, prompting the need for greater attention to vocoder impacts and stronger attack models alike.

Index Terms: speaker anonymization, automatic speaker verification, privacy

1. Introduction

The task of speaker anonymization broadly refers to the processing of speech recordings to conceal the speaker identity while preserving the linguistic and paralinguistic content. The topic has attracted increasing research interest in recent years, in particular through the VoicePrivacy Challenge [1, 2], which was founded in 2020 to define the problem, provide strong baselines, foster progress and identify research priorities. No matter what the application, anonymization should protect an appropriate trade-off between privacy and utility. Privacy can be estimated using automatic speech recognition (ASR) and a word error rate (WER) metric which reflects the degree to which linguistic and paralinguistic content is preserved.

Most anonymization solutions are based upon original work [3] and upon the extraction and processing of three different representations [4]:
- a set of linguistic features produced by an ASR model;
- a representation of intonation and prosody, usually in the form of a fundamental frequency (F0) curve;
- an x-vector, namely a neural embedding which encodes the speaker identity [5].

To conceal the speaker identity, the x-vector is typically perturbed by means of an anonymization function, thereby obtaining a new pseudo-speaker embedding. The three components are then fed to a waveform synthesis model (a vocoder) to produce an utterance in the voice of the pseudo-speaker.

The anonymization function used by two of the three VoicePrivacy baselines utilizes a pool of external x-vectors. The pseudo-speaker x-vector is derived from a subset of the furthest vectors in the pool from the input x-vector. Most VoicePrivacy participants focused predominantly upon improving the anonymization function to enhance privacy [6, 7, 8]. This focus can imply an assumption that no other processing stages contribute substantially to anonymization. We have found this not to be the case.

We report in this paper our work to observe and compare the relative impacts of a conventional x-vector anonymization function and a vocoder, two components of a state-of-the-art anonymization system [9]. We show that both components contribute to anonymization and that the contribution of the vocoder, which we refer to as the vocoder drift, is in some cases even greater than that of the anonymization function. We demonstrate that this phenomenon is also common to other popular vocoders. Collectively, they fail to provide the level of fine-grained control over the input/output x-vector space that would otherwise justify the focus within the community upon the anonymization function. Finally, we show that the vocoder drift can be learned by an attacker, knowledge which can be exploited in order to reverse the anonymization. Our findings corroborate other evidence [10] that the protection provided by such approaches to anonymization might be overestimated.

2. Relation to prior work

In this section we describe the typical, high-level structure of an x-vector–based speaker anonymization system (see Figure 1), along with relevant prior work. We then introduce our own setup which we used for all experiments reported in Section 3 and Section 4.

2.1. X-vector–based speaker anonymization

Let \( s \in \mathbb{R}^L \) be an input speech utterance of \( L \) samples. The input is first frame-blocked into a sequence of \( N \) frames and then decomposed into three separate representations comprising:
- an F0 curve \( f \in \mathbb{R}^{N} \) which is intended to encode intonation and prosody; a set of \( d \)-dimensional linguistic features \( G \in \mathbb{R}^{d \times N} \) which encode the spoken content (the text); an x-vector \( x_o \in \mathbb{R}^{m} \) which encodes the speaker identity, where subscript \( o \) denotes extraction from an original input utterance.

A vocoder model \( V(f, G, x_o) \) is trained to reconstruct input waveforms from the decomposition. Anonymization is achieved by replacing \( x_o \) with a substitute so as to conceal the speaker identity, but by using the other components unchanged in order to preserve remaining speech attributes. The substitut
tion is performed using an anonymization function \( a(x_o) = x_p \in \mathbb{R}^m \) to perturb the original x-vector. An anonymized ut-
terance \( \hat{s} \) in the voice of a fictitious, pseudo-speaker determined by
the anonymized x-vector \( x_p \), is then synthesized according to \( \hat{s} = V(f, \hat{G}, x_p) \). The anonymized utterance should maintain
the same linguistic and paralinguistic content as the original input
signal. As discussed later, an additional x-vector \( x_a \) can be extracted from \( s \) in order to measure privacy.

By convention, \( a(\cdot) \) acts to create a new pseudo-speaker
using speaker embeddings drawn from an external pool of x-
vectors \([1, 2, 3, 4, 9, 10]\). Given an input \( x_o \), the \( K \) vectors
within the pool that are furthest from \( x_o \) according to some dis-
tance metric are selected and then, from among them, \( K' \) vec-
tors are chosen randomly and averaged to obtain \( x_p \). The design of this function has received considerable attention, with numer-
ous works having investigated how its configuration, the choice
of distance metric \([11]\) and the strategy by which x-vectors are
selected from the pool \([11, 12]\) influence performance. The
participants of the two VoicePrivacy Challenges held in 2020
and 2022 proposed different enhancements to \( a(\cdot) \). They in-
clude the generation of pseudo-speaker embeddings using a
generative adversarial network \([6, 13]\) and adversarial noise \([7]\),
among others \([8, 14]\). None of the participants reported the in-
fluence of the vocoder. In this paper, we show that it too con-
tributes to anonymization and that it can be responsible for a
great deal of the privacy protection.

2.2. Our approach

Our approach is based on the pipeline described in \([9, 10]\).\(^1\) The
F0 curve is estimated using YAAP \([15]\). The linguis-
tic feature extractor is a HuBERT-based soft content encoder \([16]\)
and x-vectors are extracted using ECAPA-TDNN \([17]\). We experi-
mented with three vocoders: the HiFi-GAN \([18]\), originally
used in \([9]\); the neural source filter (NSF) model \([19]\) as used
by baseline B1a of the VoicePrivacy Challenge held in 2022; a
variation of the HiFi-GAN which uses a NSF model as genera-
tor, as used by baseline B1b of the same VoicePrivacy Challenge
edition \([2]\). We use the conventional pool-based anonymization
function \( a(\cdot) \) described above with \( K = 200, K' = 100 \), and
a cosine distance metric.

Like most related work, we use the VoicePrivacy database
and standard protocols \([2]\). The LibriTTS-train-clean-100
dataset is used for vocoder training. The LibriSpeech-test-clean
and VCTK datasets (decomposed into male and female subsets)
are used for evaluation. The external pool of x-vectors is de-
ferred using the LibriTTS-train-other-500 \([20]\) dataset. Privacy
is evaluated using ASV experiments comprising a set of enroll-
ment utterances that an attacker attempts to match to a set of

\(^1\)Code available at github.com/eurecom-asp/vocoder-drift.

<table>
<thead>
<tr>
<th>target</th>
<th>HiFi-GAN</th>
<th>NSF</th>
<th>HiFi-NSF</th>
</tr>
</thead>
<tbody>
<tr>
<td>LibriSpeech (F)</td>
<td>1.3</td>
<td>0.62</td>
<td>0.91</td>
</tr>
<tr>
<td>LibriSpeech (M)</td>
<td>1.2</td>
<td>0.56</td>
<td>0.80</td>
</tr>
<tr>
<td>VCTK (F)</td>
<td>1.3</td>
<td>0.67</td>
<td>0.92</td>
</tr>
<tr>
<td>VCTK(M)</td>
<td>1.3</td>
<td>0.59</td>
<td>0.90</td>
</tr>
</tbody>
</table>

3. Vocoder drift

In this section, we introduce the notion of vocoder drift and
report an investigation of its impact upon x-vector perturbation
and privacy.

3.1. Definition

Figure 1 shows the three x-vectors used in this work. The first
\( x_o \) is extracted from the original utterance \( s \) (left in Figure 1).
A second x-vector \( x_a \) can be extracted from the anonymized
utterance \( \hat{s} \) (right). The third x-vector \( x_p \) is the output of the
anonymization function (middle). We denote the separate do-
mains of \( x_o \), \( x_a \) and \( x_p \) as \( \hat{O} \) (original), \( \hat{A} \) (anonymized)
and \( \hat{P} \) (pre-vocoder), respectively.

As described in Section 2, the majority of research has fo-
cused on improving the anonymization function \( a(\cdot) \), the
general hypothesis being that this component is primarily respon-
sible for ensuring privacy. Intuitively, privacy is improved by in-
creasing the difference between \( x_o \) and \( x_p \), e.g. according to the
cosine distance. With the focus being upon the anonymization
function, there is an inherent, perhaps unrealistic assumption
that the vocoder preserves this distance such that the difference,
which we term as the drift, between the x-vectors at the input
\( x_o \) and that which can be extracted from the output \( x_p \)
is only modest. In this work, we seek to test this assumption.

We model the relationship between the \( \hat{P} \) and \( \hat{A} \) domains
with a function \( v(x_o) = x_a \). It allows us to define the trajectory
of an x-vector through the whole anonymization system as
\( v(x_o) : x_o \rightarrow x_a \), where \( v \) denotes function composition.

3.2. X-vector perturbation

In seeking to quantify the impact of \( v(\cdot) \) on the x-vector trajec-
tory, we define two metrics. Let \( d \) be some distance measure
over \( \mathbb{R}^m \). We then define:

\( d(x_o, x_p) \) as the target distance, a measure of how far \( x_o \)
is perturbed away from its original position according to \( a(\cdot) \);

\( d(x_o, x_a) \) as the vocoder drift, a measure of the shift between
the input x-vector \( x_o \) and that which can be extracted from
the vocoder output \( x_a \), introduced by means of \( v(\cdot) \).

Intuitively, it is desirable that \( \text{drift} \ll \text{target} \), which means
the anonymization system provides fine-grained control over the fi-
nal position of \( x_o \); it is close to the targeted pseudo-speaker em-
bidding \( x_p \). If this is not the case, then the x-vector trajectory is
determined in considerable part by \( v(\cdot) \) the x-vector extracted
from the output \( x_a \) is far from the target and the system does
not provide fine-grained control over the x-vector space in \( \hat{A} \).
Table 2: Privacy protection of the x-vector domains at different stages of the anonymization pipeline (EER, %) on test sets of LibriSpeech and VCTK, separated by speaker sex.

<table>
<thead>
<tr>
<th></th>
<th>LibriSpeech (F)</th>
<th>LibriSpeech (M)</th>
<th>VCTK (F)</th>
<th>VCTK(M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HiFi-GAN</td>
<td>0.54</td>
<td>2.51</td>
<td>15.0</td>
<td>17.9</td>
</tr>
<tr>
<td>NSF</td>
<td>0.88</td>
<td>2.99</td>
<td>14.5</td>
<td>20.3</td>
</tr>
<tr>
<td>HiFi-NSF</td>
<td>1.13</td>
<td>5.59</td>
<td>25.3</td>
<td>31.0</td>
</tr>
<tr>
<td>LibriSpeech (F)</td>
<td>0.17</td>
<td>3.04</td>
<td>18.5</td>
<td>16.7</td>
</tr>
</tbody>
</table>

We compute the average drift and target for each database subset and each vocoder: results are shown in Table 1. The target is in the order of 1.3 for all four subsets. The value of these distances lies in their comparison to estimates of the drift shown in the last three columns. For the HiFi-GAN vocoder, the drift is almost half the target distance. Lying between 0.8 and 0.97, the drift for the NSF and HiFi-NSF vocoders is substantially greater still, with drift distances almost as large as target distances. These results show that the control over the x-vector domain \( \hat{A} \) is potentially low and suggest that the x-vector anonymization and vocoder functions have an almost-comparable contribution to x-vector perturbation. It is still necessary, however, to explore their resulting impact upon privacy.

3.3. Impacts upon privacy

We follow the VoicePrivacy-defined approach to measure privacy impacts. We report a set of ASV experiments using different combinations of x-vectors. In all cases, privacy is measured using estimates of the EER. Enrollment and trial utterances are as defined by the VoicePrivacy protocol (see Section 2.2). There are several enrollment utterances per speaker. Individual x-vectors are extracted from each, averaged, and compared to a number of trial utterances. For each utterance, we extract the set of \( x_o, x_p \), and \( x_\tilde{a} \). Each set of experiments is conducted three times, with each iteration using one of the three different x-vectors. Results using the set containing \( x_o \) x-vectors (O domain) provide a baseline. Those derived from the set of \( x_p \) x-vectors (P domain) provide an indication of the contribution to privacy of the anonymization function \( a(\cdot) \). Results using final set containing \( x_\tilde{a} \) x-vectors (\( \hat{A} \) domain) provide an indication of the contribution of the vocoder function \( v(\cdot) \). Once again, we report results for the same experiment using all three vocoders.

Results are shown in Table 2, for the same database subsets as in Table 1. Baseline results for the \( \hat{O} \) domain show EERs of approximately 1%. In the \( \hat{P} \) domain, increases in the EER to between 2.5% and 5.6% indicate that the anonymization function delivers only a low level of privacy. In the \( \hat{A} \) domain, however, EERs are substantially higher for all three vocoders, if still far from providing perfect privacy (EERs of 50%). The comparison of results for \( \hat{P} \) and \( \hat{A} \) domains show that the vocoder plays a dominant role; most of the anonymization can be attributed to vocoder drift. We explored this phenomenon with t-SNE visualizations [21] of pooled x-vectors. Results are illustrated in Figure 2, which depicts a distribution of x-vectors for the male partition of the LibriSpeech dataset. In the \( \hat{P} \) domain, speaker clusters are still clearly distinguishable, while the bulk of the anonymization can be attributed to vocoder drift.

One could claim that these findings are neither surprising, nor cause for concern. There is no guarantee that the vocoder function \( v(\cdot) \) is invertible in any way which would allow the recovery of x-vector inputs \( x_p \) in the \( \hat{P} \) domain. Since the attacker does not have access to the \( \hat{P} \) domain, but only to the \( \hat{A} \) domain, whether anonymization is attributed to the anonymization function or the vocoder function is of little consequence. In the next section, we disprove these arguments and show that an attacker can learn this function or, more specifically, how to undo it. Armed with the inverse function \( v^{-1}(\cdot) \), an attacker can estimate an x-vector in the \( \hat{P} \) domain that corresponds to an x-vector in the \( \hat{A} \) domain and hence reverse the anonymization.

4. Drift-reversal attacks

In this section, we introduce drift reversal, a novel attack against anonymization systems.

4.1. Attacks on anonymization systems

Since speaker anonymization is a relatively new research topic, it is hardly surprising that little attention has been dedicated to attacks against it. Even so, the VoicePrivacy Challenge has explored the robustness of anonymization systems under a so-called semi-informed attack model [2]. Under this scenario, an attacker is aware of anonymization having been performed, and seeks to overcome it (break the anonymization) by using a similar system to generate anonymized data with which to train an ASV system. Evaluations using ASV systems trained using in-domain (similarly anonymized) data show the potential for attacks to circumvent anonymization. A more explicit approach is reported in [10] and can be used by an attacker to invert a complete anonymization system by means of a rotation matrix and to estimate speaker embeddings \( x_o \) in the unprotected domain \( \hat{O} \) from protected x-vectors \( x_p \) in \( \hat{A} \). Our approach is different since we aim to explore the anonymization robustness when we revert only the vocoder drift to recover an estimate of \( x_p \) in \( \hat{P} \).

4.2. Definition and implementation

In the case that the bulk of the anonymization performance can be attributed to the vocoder function \( v(\cdot) \) instead of the anonymization function \( a(\cdot) \), a drift reversal attack can be mounted to undo most of the protection: Let \( \hat{s}^{\theta} \) be an original (i.e. unprotected) enrollment utterance. An attacker can derive a representation of this signal in the \( \hat{P} \) domain by extracting an x-vector \( x_p^{\theta} \) and then by computing \( a(x_p^{\theta}) = x_\tilde{a}^{\theta} \). Now let \( \hat{s}^{\theta} \) be an anonymized trial utterance with corresponding x-vector \( x_\tilde{a}^{\theta} \) in the \( \hat{A} \) domain. The attacker can estimate a representation in the \( \hat{P} \) domain \( x_p^{\tilde{\theta}} \) by reversing the vocoder drift, i.e. by computing \( v^{-1}(x_\tilde{a}^{\tilde{\theta}}) \).

While the inverse function is not analytically tractable, the attacker can attempt to learn a function \( g_\theta(\cdot) \approx v^{-1}(\cdot) \) using a database of training pairs \( x_p, \hat{s} \) and anonymized utterances \( \hat{s} \). Function \( g_\theta \) can be learned using a neural network to map an anonymized utterance \( \hat{s} \) to an approximation of the corresponding x-vector \( x_p \) in \( \hat{P} \). This can be achieved by optimizing the objective function

\[
\min_{\theta} \varnothing(d(x_p, g_\theta(\hat{s})))
\]

(1)

where \( d \) is the cosine distance. Training pairs \( \{(x_p, \hat{s})\} \) can be obtained by applying anonymization to any appropriate (even unlabeled) speech dataset.

Because function \( g_\theta \) is effectively an x-vector extraction operation, we fine-tune a pretrained ECAPA-TDNN model to learn it. In line with the VoicePrivacy protocol, the model
is trained using the LibriSpeech-train-clean-360 dataset, although approximately 3% of the data is set aside for validation purposes. Still in line with the VoicePrivacy protocol, anonymization is performed at the speaker level in deriving $x_p$ for each enrollment and trial utterance, instead of at the utterance level. The network is fine-tuned for 3 epochs using Adam optimizer [22] with a learning rate of $5 \cdot 10^{-5}$ and a batch size of 8. Validation is performed every 200 iterations. Attacks are performed using the network for which the validation loss is lowest.

### 4.3. Evaluation

We compare the drift reversal attack to related VoicePrivacy lazy-informed and semi-informed attacks. For the former, the attacker compares enrollment and trial utterances which are both in the $\hat{A}$ domain, but with an ASV model trained using data in the $\hat{O}$ domain; other than by anonymizing the enrollment utterance, there is no compensation for operating upon anonymized data. The semi-informed attacker makes greater effort and uses an ASV system that is trained using an independent set of similarly-anonymized data. The latter is the default VoicePrivacy attack model.\(^3\) The lazy-informed attack is implemented using the original, pretrained ECAPA-TDNN for x-vector extraction. The semi-informed attack is performed using an ECAPA-TDNN model which is fine-tuned using AAM-softmax loss [23] and the same training settings as the drift reversal attack model.

Privacy evaluation results in terms of EER estimates are presented in Table 3 for each vocoder and each dataset. EER results for unprotected data (no anonymization) are shown in column 3 and provide a reference against which EERs for protected data can be compared. Results for the lazy-informed attack are shown in column 4 and show substantial privacy gains (higher EERs). This setting, however, gives a false sense of protection. Results for the semi-informed attack shown in column 5 show considerably lower privacy gains; by retraining the ASV function to pseudo-speaker optimization. This finding should not discourage further work in the design of x-vector anonymization functions, however. Instead, it should encourage design toward more grounded criteria.

![Figure 2: t-SNE visualization of the x-vector trajectory for LibriSpeech trial utterances (M) across the three x-vector domains (left). Focus on the trajectory of a single speaker (right). Best viewed in color.](image)

Drift-reversal attacks rely on the fact that the x-vectors fed to the vocoder, though allegedly anonymized, still have a low level of protection. This is the result of an over-deterministic anonymization function; similar x-vector outputs will produce similar x-vector outputs, thus producing trial and enrollment speaker embeddings which are close in the output domain, and thus easy to match as the same speaker, even when anonymized. That is the case for the pool-based anonymization function. Future work should investigate less deterministic anonymization functions to improve privacy directly in their output domain. Improvements to privacy in this domain will not only undoubtedly mitigate the risk of vocoder-drift-reversal attacks, but likely also that of semi-informed attacks, which might inadvertently learn to exploit the same kind of vulnerability during training.
6. References


