MDPI

Article

AudioUnlock: Device-to-device Authentication via Acoustic Signatures and One-class Classifiers

Anistoroaei Alfred[1], Iosif Patricia[2], Burlacu Camelia[3], Berdich Adriana[4] and Groza Bogdan [5]

[1] Politehnica University of Timisoara, Romania; alfred.anistoroaei@aut.upt.ro
[2] Politehnica University of Timisoara, Romania; patricia.iosif@student.upt.ro
[3] Politehnica University of Timisoara, Romania; camelia.burlacu@student.upt.ro
[4] Politehnica University of Timisoara, Romania; adriana.berdich@aut.upt.ro
[5] Politehnica University of Timisoara, Romania; bogdan.groza@aut.upt.ro
* Correspondence: bogdan.groza@aut.upt.ro

Abstract: Acoustic fingerprints can be used for device-to-device authentication due to manufacturing-induced variations in microphones and speakers. However, previous works have focused mostly on recognizing single devices from a set of multiple devices, which may not be sufficiently realistic since in practice, a single device has to be recognized from a very large pool of devices that are not available for training machine learning classifiers. Therefore, in this work, we focus on one-class classification algorithms, namely one-class Support Vector Machine and Local Outlier Factor. As such, learning the fingerprint of a single device is sufficient to recognize the legitimate device and reject all other attempts to impersonate it. The proposed application can also rely on cloud based deployment to free the smartphone from intensive computational tasks or data storage. For the experimental part, we rely both on smartphones and an automotive grade Android headunit, exploring in-vehicle environments as the main area of application. We create a dataset consisting of more than 5,000 measurements and achieve a recognition rate ranging from 50% to 100% for different devices under various environmental conditions such as distance, altitude, and component aging. These conditions also serve as our limitations, however, we propose different solutions for overcoming them, which are part of our threat model.

Keywords: smartphone fingerprinting, microphone, loudspeaker, machine learning, acoustic signature

Received:
Revised:
Accepted:
Published:

Citation: Anistoroaei, A.; Iosif, P.; Burlacu C., Berdich A.and Groza B. AudioUnlock: Device-to-device Authentication via Acoustic Signatures and One-class Classifiers. *Journal Not Specified* **2025**, *1*, 0. https://doi.org/

1. Introduction

The need for securing device-to-device (D2D) communication has significantly increased with the growth of the smart devices market, which includes smartphones, smartwatches, ear pods, augmented reality headsets and many others. This is also reflected by the over-increasing number of research papers that address this topic. It is commonly known that microphones and loudspeakers have imperfections caused by the manufacturing process, making them unique. These differences come from how the transducer responds to the sound pressure [1], as well as how the voice coil and the diaphragm manage to produce sound [2]. The chemical composition of the materials used in one production batch compared to the next, the wear of the manufacturing equipment, or variations in the electromagnetic properties of the driver give a uniqueness to each microphone and loudspeaker, as illustrated in [2]. These differences cannot always be discerned by the human ear, but signal processing techniques and machine learning algorithms can be used to detect such differences. Because of this, audio signals might be just the perfect way to deliver the right solution for D2D authentication.

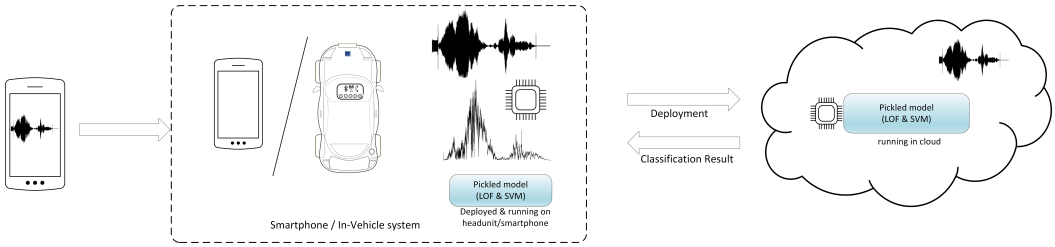


Figure 1. System overview

The fact that most modern-day devices have microphones and loudspeakers already incorporated makes this choice a natural one. In the automotive context, this concept fits even better due to the fact that headunits shifted their main usage from basic radio functionalities to a whole package of services, like audio-video streaming from the internet, providing advanced features to the users, remote control of the components, etc. Modern-day cars still use classical keys for authentication, while only a few high-end manufacturers are releasing solutions that use smartphones. But existing initiatives such as the Car Connectivity Consortium [3], where car and phone manufacturers are developing industry standards and solutions to use smartphones as car keys, are promising for the future of smartphones within the car ecosystem. So far, the main solution released by the aforementioned collaboration is based on NFC and UWB, but other options might be taken into account, a reason for conducting more research on alternative authentication sources. All of these come hand in hand with advances in headunit hardware, which are able to perform more and more complex computations.

Another specific area of application may be improving multi-factor authentication (MFA), which requires the users to provide multiple pieces of evidence (factors) in order to prove their identity. Needless to say, users still heavily rely on passwords and generally, when employed alone, these are far from being sufficiently secure (mostly because users are prone to choosing weak entropy passwords). The premise of an MFA mechanism is the following: the more factors there are, the lower the probability that an attacker forges all tokens. These multiple factors might include something the user knows (like a password, a PIN), something the user possesses (like a code generated on a mobile device, a bank card) or something that the user physically possesses (a fingerprint, face shape, voice) [4,5]. In order to avoid security breaches, most big tech companies already adopted MFA, e.g., Apple adopted MFA when checking the users' Apple ID [6], while Google and Facebook use MFA when users sign in with a new device [7,8], etc. Also, since the demand for such services is high and the implementation can be sensitive, there are even off-the-shelf solutions developed by Microsoft [9] or Google [10] for this purpose. Despite the fact that, so far, most of the industry's solutions for MFA are based on passwords or human fingerprints, the research community investigates many other factors, like audio signals, for example. Whether using audio sounds to measure the proximity of devices as in [11] and [12], or using ambient sound for authentication as in [13] and [14], there are many other papers proving that audio signals could be a reliable and effortless addition to MFA.

The physical properties of the sound make it a good candidate for security: if broadcast at a low volume, which is enough for short distances, a malicious adversary that wants to intercept needs to be present near the devices or to use more exquisite microphones. Indeed, there are several recent works showing that sound can be intercepted in clever ways, even from a significant distance, but these kinds of attacks are beyond the scope of our work [15], [16], [17], [18].

An overview of the system we design is depicted in Figure 1. Within this system framework, we focus on the usage of one-class classifiers as the main solution. This seems to be generally neglected by related works which use multi-class classification, and this

creates a disparity with the real world problem, where the task is not recognizing a single device from a larger set of multiple devices, having all fingerprints available in the dataset but rather, recognizing a single device from a larger pool of devices that are not available for training the classifiers. The target smartphone emits a synthetic sound (which has a short start frequency and afterwards a linear-sweep signal) towards another smartphone or a headunit. The recording device processes this data, filters it and interprets it (i.e., runs a machine learning model) either locally or on the cloud by sending over the normalized FFT frequencies for identification. Whether we discuss cloud or local data interpretation, we use a pickled machine learning model. This means that a Python object hierarchy (in our case, a trained machine learning model) is converted into a byte stream file [19] (pickled model or .pkl file) which can be later loaded on different environments (in our case, an Android device or a cloud server). For identification purposes, we use the Fast Fourier Transform (FFT) to extract the frequency response from the linear sweep signal (generated by the loudspeaker and recorded by the microphone), then use two One Class Classifiers: Support Vector Machine (SVM) and Local Outlier Factor (LOF). The contributions of our work are the following:

- we employed the LOF classifier which, to the best of our knowledge, was not previously used with microphone and loudspeaker data,
- we implemented an Android application that records sounds and relies on machine learning algorithms that run either on the smartphone or in the cloud, being able to differentiate between the smartphones that emitted these sounds,
- we built a dataset of 1,540 measurements using 22 different smartphones and one Android headunit in the process,
- we present concrete experimental results regarding recognition accuracy and compare two classifiers against data from multiple devices, including same model smartphones, using both a smartphone and a headunit to recognize the emitter,
- the experiments also address adversarial behavior as well as environmental variations, like temperature or atmospheric pressure changes, which can degrade classification accuracy.

The rest of this work is organized as follows. In Section 2, we present some of the related works. In Section 3, we present the implementation details alongside the devices we use. Section 4 provides the classifiers that we used.The experimental results are presented in Section 5, limitations are in Section 6, while Section 7 holds the conclusion of our work.

2. Related work

In the last few years there have been quite a few works that studied mobile device fingerprinting using different embedded sensors, e.g., from microphones or loudspeakers to cameras and accelerometers, etc. Here, we provide a brief overview of these works.

In a recent work, smartphones were fingerprinted by using the roll-off characteristics of the device's loudspeakers [20]. For higher accuracy, they also relied on deep learning algorithms such as Convolutional Neural Networks (CNN) and Bi Long Short Term Memory (BiLSTM) networks. This work was closest to the current paper but it differs in the selection of the classifiers, i.e., as already stated, the current work uses one-class classifiers which represent a more realistic choice from a practical perspective. We also tested these classifiers on the public dataset made available by [20] to check the performance of the one-class classifiers, but later relied on fresh experiments for the current paper. In contrast to [20], besides the distinct classifier, we also experimented with adversarial impersonation attempts as well as with environmental variations, i.e., temperature and atmospheric pressure.

A method to generate a stable, unique and stealthy device ID for smartphones, by exploiting the frequency response of loudspeakers, was described in [21]. For loudspeaker identification, the Euclidean distance between two samples was used and, in order to keep the process stealthy, frequencies between 14kHz and 21kHz were employed. The authors of [2] managed to fingerprint smartphones based on audio samples and considered three fingerprinting sources: loudspeakers, microphones, and joint speaker-microphone data. After recording audio data, Mel-Frequency Cepstral Coefficients (MFCCs) were used to localize spectral anomalies, while K-nearest Neighbor (KNN) and Gaussian Mixture Models (GMM) were the classifiers employed to identify the device.

There were many other works that used microphones for fingerprinting, the difference being the type of signals and the classifiers used. For example, the following research papers used synthetic sound as their signal, but employed different classifiers, like Naive Bayes (NB) classifiers, SVM and KNN in [22], inter-class cross correlation [23], Maximum-Likelihood Classification [24] or artificial neural networks [25]. Human speech was also used as fingerprinting input for different classifiers like MFCC [26], Linear Predictive Coefficients (LPC), Multi-layer Perceptron classifier (MLP) [27], Gaussian Mixture Model (GMM) [28] or Linear Frequency Cepstral Coefficients (LFCC) [29]. Some of these works used preexisting datasets, while others managed to create their own.

A source smartphone identification system suitable for both clean and noisy environments was proposed in [30]. This system used the spectral distribution features of the constant Q transform domain and four classification techniques. The authentication of smartphones using the intrinsic physical properties of the mobile smartphones' microphone using three classifiers, i.e., KNN, SVM and CNN, in the presence or absence of noises was studied in [31]. Three types of noises were considered: the Gaussian White noise, the Babble noise and environmental noise from the street. The authors in [32] fingerprint 16 identical and 16 different smartphone microphones based on several in-vehicle noises, i.e., car horn, wipers sound, hazard light sounds, locomotive and barrier sounds. They also added several types of noises, i.e., traffic and market noises.

In [33], the authors used a few multi-class classifiers for source smartphone identification based on encoding parameters of multiple audio files. Also, in [34], smartphone identification from recorded speech signals was elaborated with the goal of extracting intrinsic traces related to the smartphone used to record the data. Investigation of the self-localization problem of an ad-hoc network of randomly distributed and independent devices in an open-space environment with low reverberation but heavy noise was done in [35]. The authors used acoustic emissions and compute the time of arrival for sounds to determine the distance between two smartphones while landmark-based audio is used for synchronizing audio recordings. In [36], different audio feature extraction algorithms, which aid in increasing the accuracy of identifying environmental sounds, were evaluated. Sound signals were detected using standard deviation of normalized power sequences based on zero crossing rate, MFCC, spectral flatness or spectral centroid were applied. Other papers have also proposed methods for defending against loudspeaker fingerprinting by enabling users to control the frequency response and modify the given stimulation signal [37].

Fingerprinting was also studied based on other smartphone sensors such as cameras, accelerometers, gyroscopes, and others. For example, different works use distinctive features of the camera, like color filter array (CFA) [38], dark signal non-uniformity (DSNU) [39] or photo response non-uniformity (PRNU) [40]. Also, accelerometers were used to fingerprint devices in [24], [41] and [42]. The possibility to fingerprint devices based on combining sensors was suggested in many papers, like [43], which uses 7 sensors (acceleration, magnetic field, orientation, gyroscope, rotation vector, gravity, linear acceleration)

or [44] and [45] where 2 sensors have been used (accelerometer and gyroscope). In other papers, software characteristics like side-channel features from network traffic from several popular applications [46] or TCP [47] was used. Last but not least, magnetic signals emitted by CPU proved to also be able to fingerprint devices [48].

3. Implementation Details and Devices

In this section we discuss the implementation details related to the Android applications, how we use the cloud computational capability and a few statistical results regarding the recordings.

3.1. Microphone and loudspeaker details

Loudspeakers and microphones contain many components made of various materials. Therefore, due to imperfections in the manufacturing process, each loudspeaker and microphone will display slightly perceptible differences, just enough to make unique fingerprinting possible. Loudspeakers are electroacoustic transducers that convert electrical impulses into sound. Microphones do the opposite operation, converting sounds into electrical impulses. The vibration of a diaphragm is used to produce sound by a loudspeaker. The diaphragm is usually manufactured in the shape of a cone or dome, using different materials like paper, plastic or metal. For loudspeakers, it turns the vibrations of the voice coil into sound waves, while for microphones, it turns the sound vibrations into movement of the voice coil.

3.2. The Android Application and Devices

To prove our concept, we developed two Android applications, one that handles sound recording and one that handles emission. Afterwards, we use the Python pickle module [19] to serialize the classifiers and extend the applications so that the models can run on smartphones. Further, we also managed to expand the functionality of the applications by connecting to a cloud server and running the classifiers on the cloud.

Figure 2 shows a flowchart that describes both the recorder and emitter applications. In the emitter, we specify a start frequency that the recorder later uses to detect whether transmission has started. Subsequently, the emitter starts sending the audio signal for 5 seconds. If the recorder accepts the authentication, the emitter stops, otherwise the procedure is repeated. The recorder application is slightly more complex : in the user interface, the start frequency is specified (it can be selected by the user but must match the frequency of the emitter application). When the start frequency is detected, the raw values are added to an array for a duration of 5 seconds. Afterwards, the FFT is computed, and the values are normalized. These values are either sent to the cloud classification model or locally processed by the pickled model. A response is returned, indicating whether the smartphone was recognized or not. If the smartphone is recognized, the cloud classification model is regenerated to account for the newly collected data. Adding new data to the classification model is needed, since loudspeaker parameters may vary in time and the smartphone will become unrecognizable after longer periods (we will later show through experiments performed several months apart that this is indeed the case).

As an additional note, the sampling rate of the Android audio library is 44.1kHz. Due to the Nyquist-Shannon sampling theorem, the maximum frequency that can be perfectly reproduced is 22.05kHz. Alternatively, the sampling rate can also be set to 48kHz, but we obtained better results for the former. In order to record signals, we used the Android AudioRecord class. For the Fast Fourier Transform (FFT), we used the JTransforms library [49]. The description of the FFT function, also implemented in this library, can be expressed in a mathematical way as follows: $X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\,2\pi kn/N}, \quad k = 0, 1, \ldots, N-1$, where

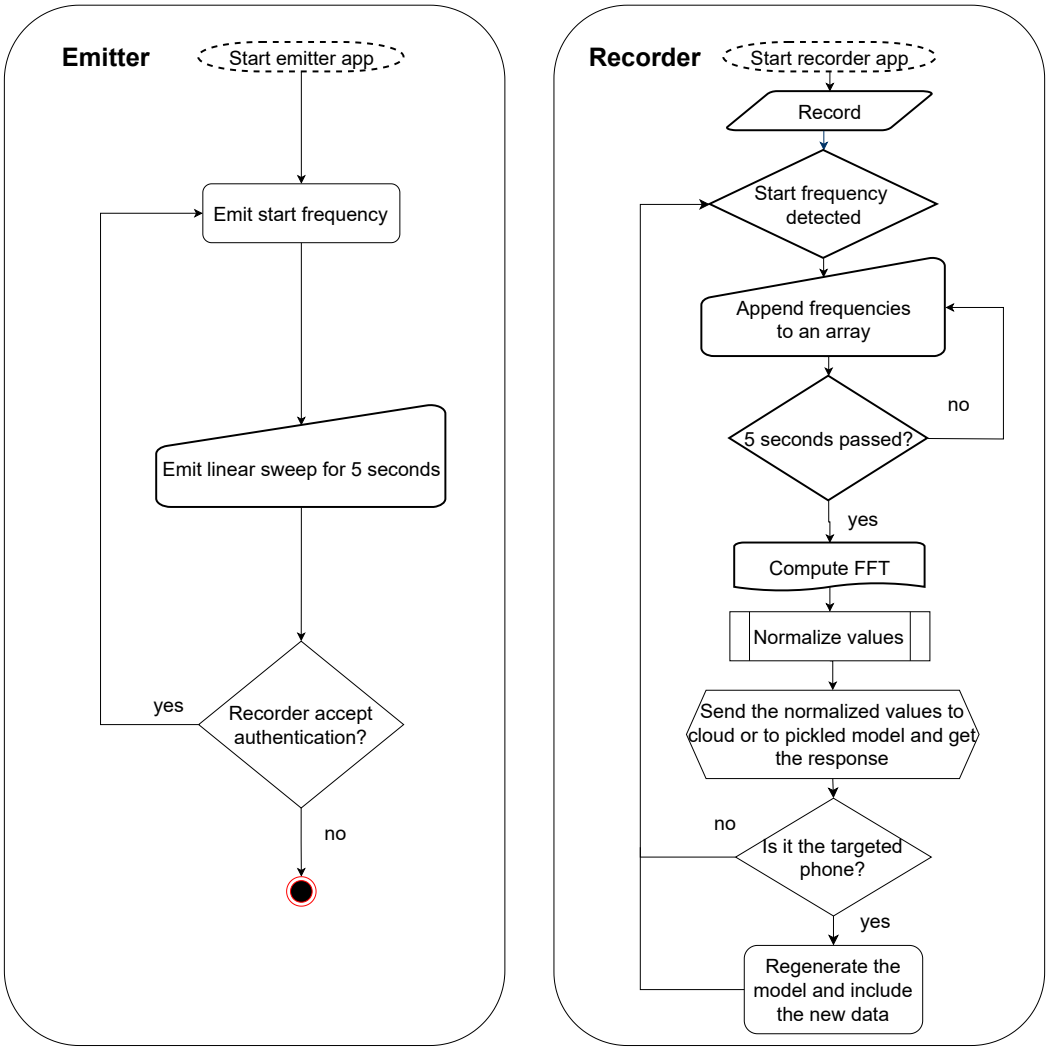


Figure 2. Flowchart of the emitter and receiver apps

$x[n]$ are the time-domain samples, $X[k]$ are the frequency domain samples, N is the total number of samples, j is the index for the time domain, k is the index for the frequency domain, and j is the imaginary unit. We parsed the resulting FFT at intervals equal to 1Hz and used only the highest amplitude in each interval. Thus, we get an array of 22,050 values.

In order to have more conclusive experiments, we used 22 smartphones from different brands and one Android headunit. We encountered problems with the headunit's incorporated microphone, which was of poor quality. We tried to use an external microphone that can be acquired for the unit, but the results did not improve at all. Therefore, we decided to stick with the internal microphone for our experiments. In Table 1, we depict the list of devices, including their manufacturer, model, label used in this work and the number of devices of the same type. In order to increase the number of devices that we are using for measurements, we decided to add several iPhones as well. Since these smartphones do not run Android, we manually added the sound to be played on each device. For this, we used REW [50] for generating the desired frequencies, i.e., the linear sweep from 0 Hz to 20 kHz and the start frequency (this was set to 12 kHz for the headunit and 16kHz for the smartphones). This linear sweep signal can be described by the following mathematical expression $f(t) = f_0 + \frac{(f_1 - f_0)}{t_1} t$. Where t denotes time, f_0 is the start frequency at time 0 and f_1 the instantaneous frequency at time t_1. Concretely, in our implementation we used

Table 1. Devices from our experiments

Brand	Model	Label	No. of devices
Allview	Viper V1	AL1	1
Apple	iPhone 5s	A1 & A2	2
	iPhone SE1	A3	1
	iPhone SE2	A4 & A5	2
	iPhone 8	A6	1
	iPhone 13 mini	A7	1
Asus	Nexus	AS1	1
Erisin	PX5	E1	1
LG	Stylus 2	L1	1
Motorola	E6Plus	M1	1
OnePlus	7T	OP1	1
Samsung	Galaxy A3	S1	1
	Galaxy A10	S2 & S3	2
	G386F Galaxy Core	S4	1
	Galaxy J5	S5 & S6 & S7	3
	Galaxy S6	S8	1
	Galaxy S7 Edge	S9	1
	Galaxy Tab S7	S10	1
		Total	23

$f_0 = 0$Hz, $f_1 = 20$kHz, $t_1 = 10$s. This led to a `wav` file which can be played on iPhones as well in order to check the identification rate from the recorder.

In order to have a more scalable system, we decided to extend it with cloud processing features. Concretely, we run the classification model on a cloud server provided by Heroku [51], while the smartphone (the recording application) was used only to collect data and display the result. The receiver application, after extracting data with the algorithm described before, connects to the cloud server using the Volley HTTP library [52] developed by Google. Data is encapsulated in a JSON object sent through a POST request. On the cloud side, Heroku offers the possibility to be configured through a Github deployment pipeline. Using Flask [53], we managed to run our pickled model in the cloud and return a response to the client (i.e., the receiver application on the smartphone), indicating whether the data belongs or not to the target smartphone.

3.3. Preliminary statistical results regarding the recordings

In order to see if there are significant statistical differences between the devices, we choose to extract several spectral statistical characteristics. For each device in our experiments, we perform 50 measurements and extract the mean value , which we present as a box plot in Figure 3. The characteristics we have taken into consideration are:

- the power of the signal in a selected frequency band (Powerband) - refers to the average power of the input audio signal in the 20Hz-20kHz band,
- the standard deviation of the spectrum (Std) - how much the frequencies deviate with respect to the mean value,
- the peak of the signal (Pks) - the maximum values of the input audio signal,
- the average value of the signal (Mean) - more precisely, the average value of the power spectrum of the audio signal
- the spikeness or flatness of the distribution compared to the normal distribution (Kurtosis) - represents a statistical parameter that quantifies the distribution shape of the audio signal compared to the Gaussian distribution, i.e., saying if the distribution is sharply peaked or not,

- the ratio of skewness to standard deviation (Skewness) - which represents the asymmetrical spread of the audio signal about its mean value.

Figure 3 shows the aforementioned statistical coefficients when a smartphone was used as the receiver (device M1), while Figure 4 shows the same characteristics when a headunit was used as the receiver (device E1). From the figures, it is clear that the devices can be easily distinguished and, moreover, the recordings performed with the headunit show larger boxes in the plot which are due to a less accurate measurement. Indeed, as the selected classifier later suggests as well, the confusion rate between devices is higher with the headunit because of the poor performance of the microphone.

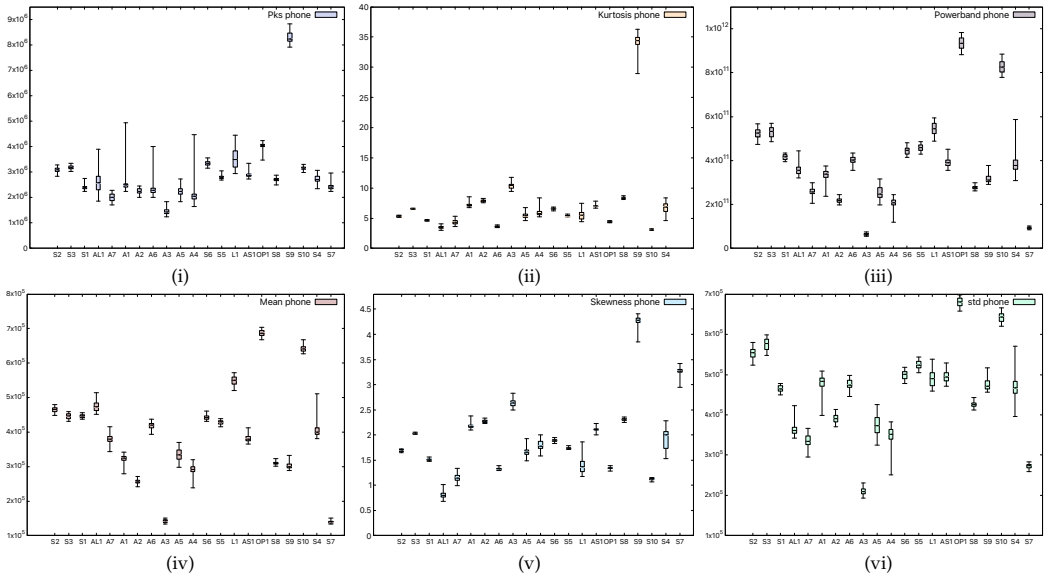


Figure 3. Statistical characteristics: (i) signal peak, (ii) kurtosis, (iii) powerband, (iv) mean, (v) skewness, (vi) standard deviation, smartphone (device M1) records

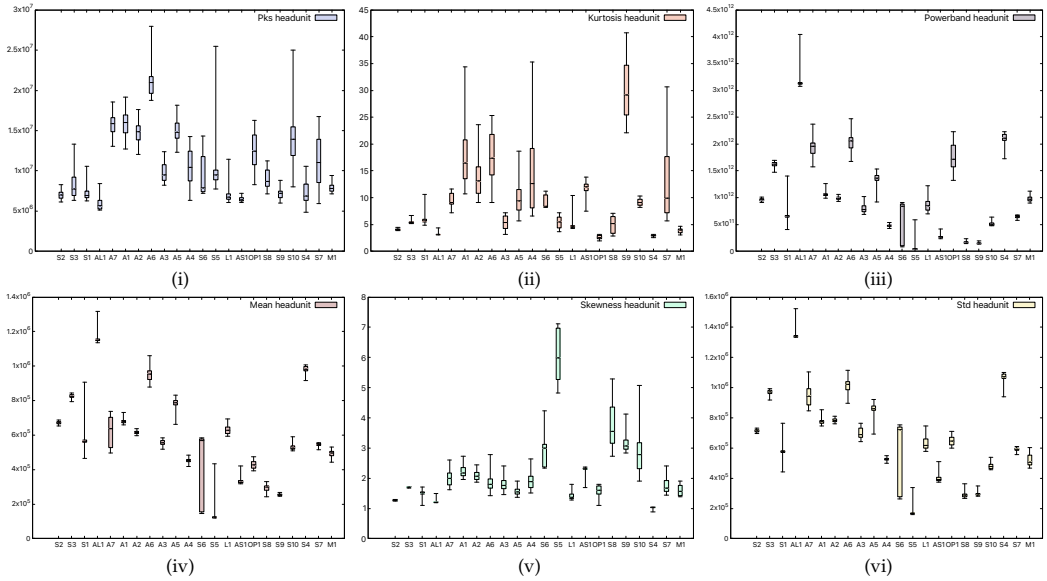


Figure 4. Characteristics extracted: (i) signal peak, (ii) kurtosis, (iii) powerband, (iv) mean, (v) skewness, (vi) standard deviation, headunit records

4. Classifiers and Noise Resilience

In this section we detail the classifiers that we used, along with some preliminary results on an existing dataset. We also consider the impact of noise on these classifiers, as noise will be an important factor in practical scenarios.

4.1. Selected classifiers

The problem we are trying to tackle is the use of one-class classification, which allows us to learn one specific device and later test the model against the other devices. As stated, most existing works use multi-class classification techniques, where all devices from the dataset are used in both training and testing phases. This approach is not sufficiently realistic and, to improve this, we decided to use one-class classification techniques, namely two such classifiers, as described next:

4.1.1. One-Class SVM (OC-SVM)

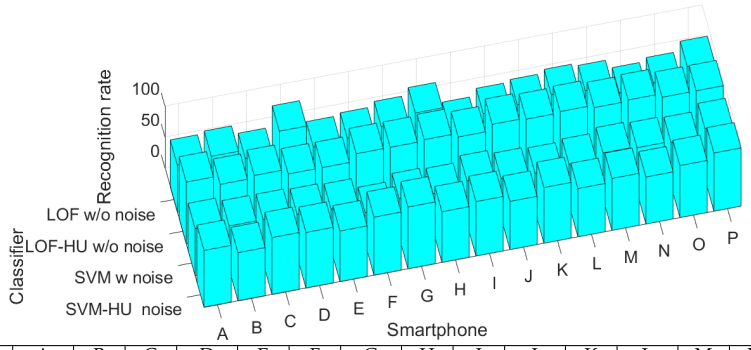
OC-SVM [54] is an unsupervised algorithm for learning a decision function for novelty detection, first introduced in [55]. SVMs do not represent a probability distribution. Instead, their approach is finding a positive function to model regions with a high density of points, and a negative function for low-density regions. All data points are separated from the origin in the feature space using hyperplanes, and the points are projected to a higher dimensional space. In contrast to typical SVMs, it employs a parameter to set the outlier proportion in the data, maximizing the distance between the hyperplane and the origin. Outliers are points that are below the hyperplane and closer to the origin. The absence of target labels through the training phase for OC-SVM is another distinction.

OC-SVM requires the following parameters to be specified: `tol` (the tolerance needed to stop), `kernel` (the type of kernel that will be used by the algorithm), `degree` (the degree of the polynomial function), `nu` (higher and lower bounds on the percentage of margin errors and, respectively, the percentage of support vectors in relation to the total number of training samples). We chose: `kernel='rbf'`, `degree=5`, `tol=1e-7` and `nu=0.001`. A kernel is required because it allows the projection of samples from the original space to the characteristics space. Radial Basis Function (RBF) is one of the most used kernels in the literature. It aids the algorithm in determining the radius of the hypersphere, as well as the polynomial functions. We used a large degree to make the algorithm more precise. The tolerance parameter, `tol`, and the `nu` parameter are set to such low values to reduce the possibility of running into many incorrect predictions.

4.1.2. Local Outlier Factor (LOF)

Local Outlier Factor (LOF) [56] is another unsupervised algorithm, designed for outlier detection, which was first introduced in [57]. It also has density at its core and aims to identify local outliers, i.e., points that are not fundamentally different from the rest of the population but deviate significantly from their neighbors. As far as density is concerned, this means that a local outlier is an example that greatly differs in density from surrounding points. The algorithm can be configured using the following parameters: the number of neighbors considered for each sample, the distance metric used, the contamination ratio of the data, as well as a boolean value that controls whether the algorithm is being used for novelty detection.

For each point, the algorithm computes a score (the local outlier factor), i.e. a degree that quantifies how much a point resembles an outlier [57]. Generally, a sample is considered an outlier if its corresponding LOF is above a certain threshold. The score itself is determined by looking at the local reachability distances (LRD) of the point of interest versus that of its K-nearest neighbors [57]. To define the LRD, a brief understanding of



%	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
LOF w/o noise	76	80	66	100	64	68	76	88	54	66	70	76	72	58	64	80
SVM w/o noise	100	86	90	82	82	94	94	96	90	100	98	100	96	98	92	92
LOF w noise	76	70	70	68	66	50	62	60	70	64	62	66	74	72	70	82
SVM w noise	92	78	92	90	82	94	100	82	88	78	90	78	84	76	84	94

Figure 5. Acceptance rate for each smartphone as bar charts (up) and numerical values (down) using the existing dataset

K-neighborhoods is needed. A K-neighborhood contains the points that lie within a circular region whose radius is equal to the distance between the point of interest and its K-th nearest neighbor (i.e., the K-distance). Similarly, the reachability distance can be defined either as the K-distance for points that lie within the K-neighborhood or, the actual distance between the point that defines the neighborhood and a target point, for samples outside of this region. The LRD of a point is then computed by averaging the reachability distance over the points in its K-neighbor, and subsequently taking the inverse of this average.

As was the case for One-Class SVM, we use the scikit-learn implementation [58] [59], choosing the following parameters: `n_neighbors=19`, `metric='manhattan'`, `contamination=0.3` and `novelty=True`. Since we are using the algorithm for novelty detection, training is done entirely on samples belonging to one known class. The literature implies that in this case, the training data should be deemed completely uncontaminated [60], however, our experiments suggest that setting a low level of contamination, i.e., `contamination=0.3`, improves the overall performance on our datasets. This could be explained by the fact that, during data collection, a certain level of environmental noise may be introduced into the samples. As for the metric used, i.e., the Manhattan distance, this was preferred for our high-dimensional data since it sums over the absolute value of the difference between each scalar, i.e., $\sum_{i=1}^{n} |x_i - y_i|$. Other works also recommend it for high-dimensional data [61]. The number of neighbors was empirically determined.

4.2. Testing classifier resilience against background noise

In order to test the reliability of the classifiers in the context of device fingerprinting, we decided to use a public dataset which contains audio signals from 16 Samsung Galaxy J5 loudspeakers with and without noise [20]. Concretely, the type of noise is Additive white Gaussian noise (AWGN) overlapped using Matlab.

The first use case tested was audio signals without noise. For this, we trained the classifiers with half of the available data for each device (50 samples) and tested them on the other half. The results are presented in Figure 5 and, as can be seen, the identification success rate is between 54%-100%. Afterwards, we focused on noise-affected data. We trained the classifiers with half of the noise-affected samples and tested them against the rest. The results are also presented in Figure 5. As can be seen, the identification success rate is between 50%-100%.

In both cases (with and without noise), the SVM classifier provides better results than the LOF classifier. For example, SVM has an average recognition rate for both cases

Figure 6. Smartphone and headunit placement

around 89%, while for LOF, this value is 70%. Comparing the classification results with and without noise, we observe that the noise influence is the same on both classifiers. For example, the performance of the LOF classifier decreases from an average value of 72% (without noise) to an average value of 67% (on data with noise). This means that 5% is lost for the identification rate when noise is added. For SVM, the performance decreases from an average of 93% to an average of 86% when adding noise to the data, which equates to a 7% decrease in the identification rate. To sum up, both one-class classifiers perform well, but SVM is better on existing datasets.

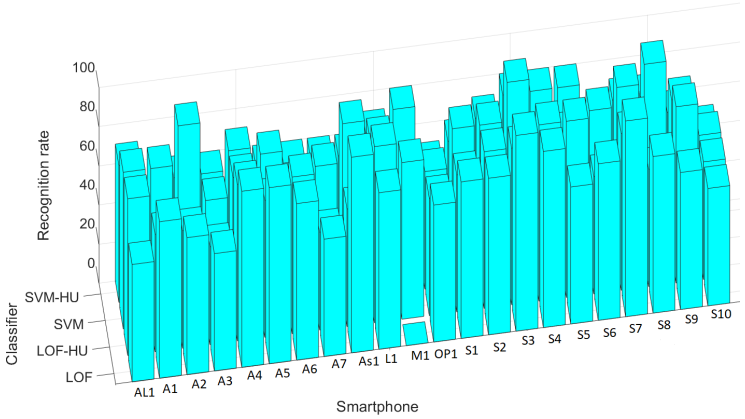
5. Experimental Results

In this section we present the experimental methodology along with various details, such as the placement of devices and the influence that distance has on measurements.

5.1. Off-line measurements and results

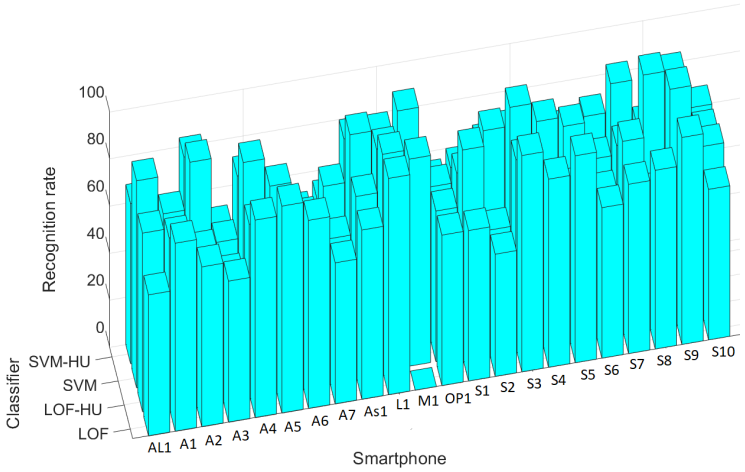
We tested the classifiers based on pre-recorded data stored in the frequency domain as FFT coefficients. We performed 50 measurements with each smartphone, having another smartphone as the recorder (in this case, M1). A measurement consists of detecting the start frequency and then the linear sweep that follows for 5 seconds; it contains all frequencies in the range of 0 Hz to 20 kHz. During the experiments, the smartphones and headunit were respectively placed at a distance of 10 centimeters, with their loudspeakers facing the microphone of the recorder. This is also shown in Figure 6, which depicts one of our smartphones during the recording process.

Of the 50 measurements, 25 were used to train the classifiers and generate the pickled model. Then, we evaluated each model on the other 25 measurements. The results are presented in Figure 7. As a side note, in the corresponding table, N/A marks the measurements that were impossible to perform because the recording and the emitting device were the same (i.e., device M1). The lowest recognition rate is 52%, while the highest is 92%. These results are influenced by many factors, but the microphone quality is the most critical. This can be observed easily in Figure 7, since the measurements done with the headunit are slightly worse than those performed with the smartphone. Indeed, the average recognition rate for the smartphone recordings is 75.80% (SVM and LOF), while for measurements



%	AL1	A1	A2	A3	A4	A5	A6	A7	As1	L1	M1	OP1	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
LOF	60	80	70	60	90	90	80	60	100	80	N/A	70	80	80	100	90	70	80	100	80	70	60
LOFHU	80	60	60	50	80	70	80	50	60	90	80	60	70	80	80	90	90	60	80	60	90	60
SVM	80	80	100	60	70	80	60	70	90	80	N/A	60	80	80	100	70	70	80	90	100	80	60
SVMHU	70	50	60	60	70	60	60	60	60	70	80	50	60	70	80	80	80	60	70	70	60	50

(i) Results for 10 measurements with classification performed in the cloud



%	AL1	A1	A2	A3	A4	A5	A6	A7	As1	L1	M1	OP1	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
LOF	60	80	68	60	84	88	80	60	72	92	N/A	64	64	52	92	80	88	64	72	76	88	64
LOFHU	76	72	64	52	72	64	64	60	76	92	88	64	88	52	80	64	64	60	80	56	56	72
SVM	88	72	92	56	88	76	60	72	92	84	N/A	64	68	84	92	84	80	64	68	96	88	68
SVMHU	68	52	84	52	72	60	52	56	80	80	88	56	60	68	72	64	68	72	84	64	84	68

(ii) Results for 50 measurements, 25 used for train and 25 for test with classification performed on the local device

Figure 7. Identification percentages for each smartphone as bar charts and numerical values

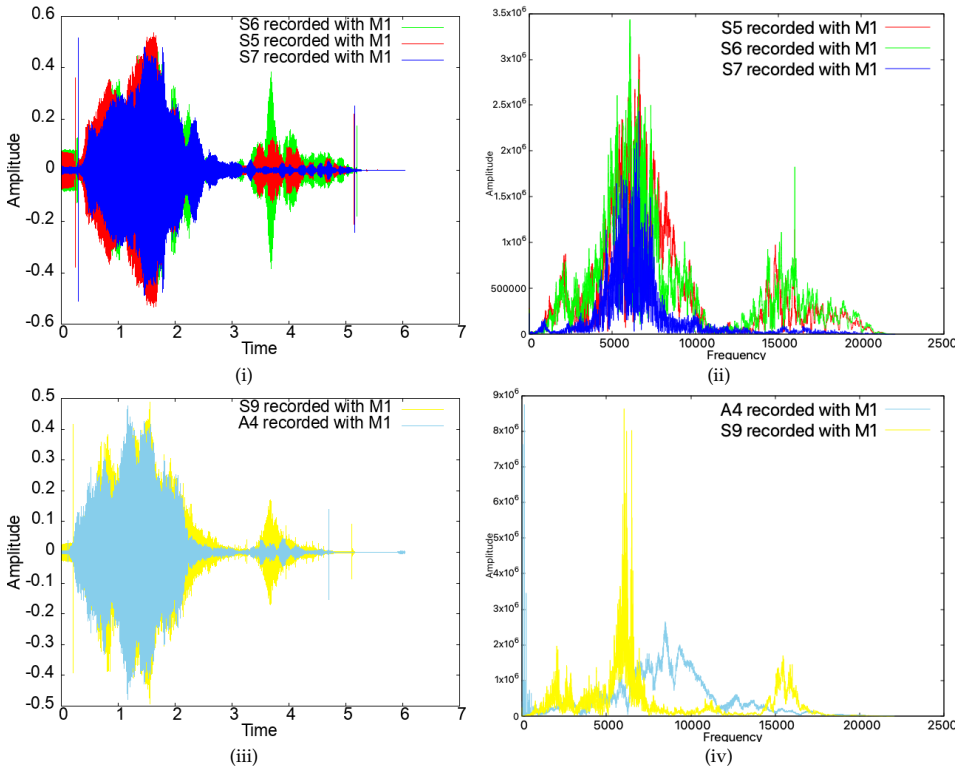


Figure 8. Time and frequency domain representations of the sweep signal for three identical devices (i), (ii) and for two different devices (iii), (iv)

performed with the headunit, the average percentage decreases to 69.36%. By comparing the classifiers, we observe that in cases when the recordings were performed with the smartphone, the SVM has slightly better results than LOF, having 77.90% average recognition rate compared to 73.71%. The difference between these two classifiers decreases when the recordings are done with the headunit, due to the poorer quality of the microphone, i.e. 70.36% for LOF and 68.36% respectively for SVM.

We note that the signals emitted by the smartphones were quite distinct, even for the same type of devices. For example, as can be seen in Figure 8, for the same smartphone model (devices S5, S6, S7 which are all copies of the same Samsung Galaxy J5), the emitted sound was quite different in both time and frequency domains. For different models (S9 and A4), the differences are even more obvious. Such differences also prevent impersonation attacks, as we discuss later in the paper.

5.2. Live measurements and results

We now test the classifiers based on live attempts to authenticate the smartphone to the headunit. For this, we run the classifiers directly on the smartphones or in the cloud, using the pickled models. All operations mentioned above, especially on the recorder app side, were not time consuming. Concretely, the computational costs introduced by each step are as follows: i) write output to a .wav file (required for off-line analysis only) 0.12 seconds, ii) perform FFT: 0.39 seconds, iii) write FFT normalized values to file: 0.58 seconds, iv) response from the model running on the smartphone: 0.01 seconds, v) response from the model running on the cloud server: 1.23 seconds. These measurements were computed as the average values after 10 attempts.

To sum up, after finishing emitting the linear sweep (5 seconds), the output (if the smartphone is recognized or not) was provided in about 1.6 seconds if the model was run locally (on a smartphone) or about 2.5 seconds if the cloud server is involved. The delays were higher for the cloud-based solution because of network timings. However,

Table 2. Success rate for 10 measurements with classifiers running on Smartphones/Headunit

Nr.	Device	LOF Phone	LOF Head-unit	SVM Phone	SVM Head-unit
1.	AL1	60%	80%	80%	70%
2.	A1	80%	60%	80%	50%
3.	A2	70%	60%	100%	60%
4.	A3	60%	50%	60%	60%
5.	A4	90%	80%	70%	70%
6.	A5	90%	70%	80%	60%
7.	A6	80%	80%	60%	60%
8.	A7	60%	50%	70%	60%
9.	AS1	100%	60%	90%	60%
10.	L1	80%	90%	80%	70%
11.	M1	-	80%	-	80%
12.	OP1	70%	60%	60%	50%
13.	S1	80%	70%	80%	60%
14.	S2	80%	80%	80%	70%
15.	S3	100%	80%	100%	80%
16.	S4	90%	90%	70%	80%
17.	S5	80%	60%	80%	60%
18.	S6	100%	80%	90%	70%
19.	S7	70%	90%	70%	80%
20.	S8	80%	60%	100%	70%
21.	S9	70%	90%	80%	60%
22.	S10	60%	60%	60%	50%

the cloud solution may facilitate more complex algorithms and a centralized solution for authenticating various devices. The cloud may also make our solution more scalable by supporting older devices or Internet of Things (IoT) with low processing power. Also updating or retraining the machine learning models would be easier and more efficient in the cloud.

To check whether the authentication results of the classifier executed on smartphone/cloud are similar to those obtained locally in the previous section, we performed another set of 10 measurements involving all the smartphones. The results are presented in Table 2. By comparing the classifiers, we show in Figure 7 that the SVM and LOF classifiers have almost the same recognition rate, concretely, 78.57% for LOF and 78.09% for SVM (note that in this case, fewer smartphone-based experiments were conducted). With the headunit, the LOF classifier has 71% average recognition rate, while SVM drops to 64%. This is in contrast with the results obtained when using off-line measurements, where the SVM classifier showed better results than the LOF classifier.

6. Limitations and discussion

In this section, we introduce the threat model and discuss the main limitations of our system.

6.1. Threat model: device impersonation by replaying sounds

The main adversaries that we consider are external devices that are able to record and attempt to reproduce the sound emitted by the legitimate device. Successfully penetrating the system would grant the adversary complete access to the services provided to the targeted user. We consider that devices that are too far from the system are less likely to produce good recordings so the distance that we take into account for the adversary is below 2 meters. Consequently, we also tested the application against a targeted impersonation attack. For this, we recorded the frequencies emitted by several devices and we played them from other devices to check if any device impersonation occurs. Such impersonation attempts consistently failed due to significant differences between the recorded sound and the original signal. For example, as shown in Figure 9, when we record the sound emitted by device S1 with device S7 and later play the recording, we can see that the amplitudes are significantly higher than those of the original signal. Also, when we record

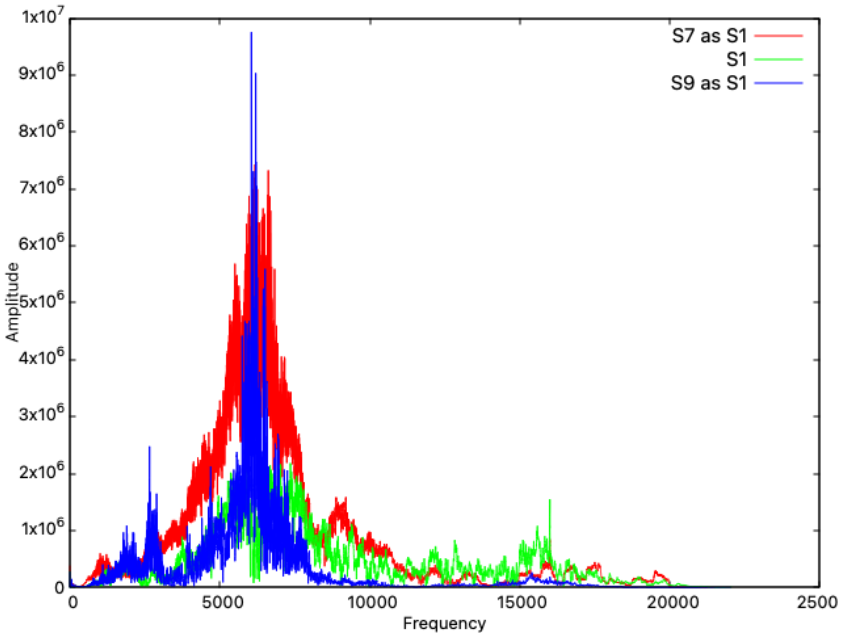


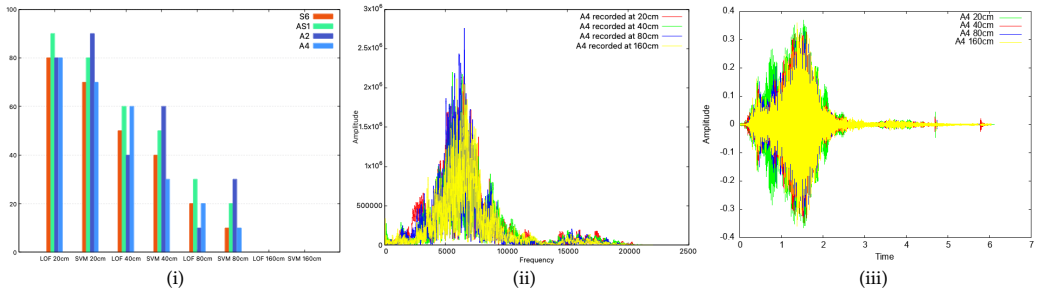
Figure 9. Frequency domain representation in the case of S1 impersonation

the sound emitted by device S1 with device S9 and later play the recording we can see that the signal pattern is completely different from the original one, despite the amplitudes being almost the same. Since the overall shape of the signal is different, it is very easy for the classification algorithms to repel the impersonation attack.

6.2. Distance influence on the recognition rate

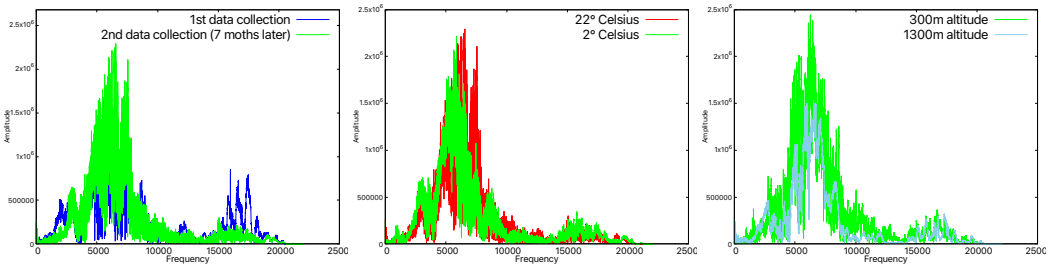
It is obvious that the distance between devices has an impact on the identification accuracy. The authors of [62] also provide rigorous equations for the acoustic attenuation, exhibiting a power-law, better described by: $S(x + \Delta(x)) = S(x)e^{-\alpha(\omega)\Delta(x)}, \alpha(\omega) = \alpha_0\omega^\eta$. Where S is the amplitude of an acoustic field variable such as velocity or pressure, $\Delta(x)$ is wave propagation distance, $\alpha(\omega)$ the attenuation coefficient and ω the angular frequency, while η and α_0 are material dependent parameters empirically obtained.

To test the impact of distance, we used 4 devices: S6, AS1, A2, A4. The first thing we did was collect 10 samples for certain fixed distances between devices, i.e., 20cm, 40cm, 80cm and 160cm. In addition, we tested two scenarios. Firstly, both classifiers (LOF and SVM) were trained with half the samples and tested on the other half. In this case, the recognition rate did not decrease with the distance, but rather, it remained mainly steady for all distances. The results are similar to the case referenced in Figure 7 (i) and comparable with those in Figure 7 (ii). This can be explained by the fact that, if we retrain the classifier with new audio signals affected by the distance, it will recognize signals originating from the same distance. Secondly, we used the classifier already trained with data from Subsection 5.1 and we observe that the recognition rate decreased from over 70% for both classifiers with devices placed at 20 cm to 0% when the devices were moved 160 centimeters apart, as can be seen in the bar-charts from Figure 10 (i). This is expected, since the audio signal is affected by the distance and thus the device is harder to recognize. In practice, we consider this distance more than enough to pair two devices (for example, the distance inside a car between the microphone and any passenger's smartphone will be less than 160 cm, if we take into account the automotive scenario). In Figure 10 (ii) and (iii) we also present plots in both the time and frequency domains to show the impact of distance.



(i) (ii) (iii)

Figure 10. Identification rate along to the distance as bar charts (i), distance influence, frequency domain for the A4 smartphone (ii), distance influence, time domain for the A4 smartphone (iii)



(i) (ii) (iii)

Figure 11. Time domain representation for 7 months time difference for A4 smartphone (i), time domain representation in the case of different temperatures for A4 smartphone (ii), time domain representation in the case of different altitudes for A4 smartphone (iii)

6.3. Environment influence on the recognition rate

Device aging affects both microphones and loudspeakers, as well as any other component, thus resulting in slight changes in the audio waveform produced/captured by the smartphone. Since microphones and loudspeakers are electromechanical devices, that is, they have moving components whose elasticity changes with time (e foam or diaphragm), they are even more affected than simple electronic components. We observed this empirically too, by using the machine learning models with data obtained 7 months apart. In this case, the smartphones were not recognized anymore, that is, data collected after 7 months from the original training data no longer matched. Figure 11 (i) shows the differences in the recorded data for the A4 device after 7 months. By reducing this interval to 1 month after training, all devices were recognized. This justifies our choice, already discussed in the flowchart of our application, to add each new sample that was correctly identified to the training dataset. After longer periods of inactivity, in practical scenarios, a user will likely have to rely on other authentication factors in order to update the fingerprint of the smartphone or to reauthenticate the device. In fact, this recommendation is already included in some security standards or draft recommendations from the industry, such as NIST SP 800-63B [63], OWASP [64], ISO/IEC 27001 [65] and ISO/IEC 27002 [66], etc. In this context, our cloud component proposal can help by regenerating the model each time a smartphone is recognized. The training dataset will grow with time, but the cloud can handle it and will contain latest signals.

Beside aging, there are also other factors that influence the device's components. For this, we verified if lower temperatures can also affect the results. Therefore, the smartphones were placed in a refrigeration environment for about 30 minutes and we tested the behavior at various temperatures. At 8°C, 14°C and 18°C, the devices were easily recognized. However, when we lowered the temperature to 2°C the models were not able to recognize the smartphones anymore. At this temperature, it can be seen in Figure 11 (ii) that the waveform is quite different to what we recorded at room temperature. This suggests that if high temperature variations are to be expected, then dedicated models

have to be trained. Some smartphones, depending on how the Linux kernel is configured, provide access to the thermal zones temperature files [67] which contain the temperature of the smartphone's CPU. This can be used to enable specific models according to the CPU temperature, but it goes beyond the scope of our work.

Another factor that might impact the recognition rate is the atmospheric pressure. The first measurements were done by us at about 300 meters above sea level, and we managed to perform another round of measurements at about 1,300 meters above sea level. The recognition rate was the same, so a variation of 1,000 meters in altitude, which corresponds 1.56 psi, had no impact. The signal remained almost identical in both cases, as can be seen in Figure 11 (iii). According to [68], altitudes above 3600 meters have a noticeable effect on sounds, caused by atmospheric pressure. We were unable to conduct experiments in such conditions since we did not have a hyperbaric chamber at our disposal.

The calibration of models for variations caused by aging, temperature, or pressure requires larger amount of data and processing time; thus, the cloud extension of the proposed system facilitates rapid regeneration and testing in the cloud. We evaluated the computational time and obtained an average of 13.5 seconds when using Google Colaboratory [69] for training on approximately 550 samples. The first run, during which the necessary packages were loaded and the drive was mounted took 71.5 seconds.

7. Conclusions

Our results indicate a good recognition rate for smartphones, with an average of 77.90% for the LOF algorithm and 73.71% for SVM, when running on smartphones, but this may be influenced by multiple factors. One such factor is the quality of the microphones: while for most smartphones the microphones were of good quality, the Android headunit had a decreased recognition rate due to the poor quality of its microphone. This issue can be mitigated by using an external microphone of better quality. Another factor that impacts the results is the distance between microphones and loudspeakers. We performed most experiments with the devices placed 10 centimeters apart and increased the distance up to 160 centimeters, until the microphone was no longer able to record the start frequency. Given the nature of the scenario, it is easy for users to place the devices close to one another, so distance should not be a concern. Clearly, a bigger concern is the aging of the device and the effect of temperature or even altitude that we discussed in the last section. In this respect, the most significant concern, show through our results, is aging, as measurements taken several months apart may lead to serious classification errors. Continuously updating the device fingerprint seems to be the only means to circumvent this issue.

Regarding the 2 algorithms that we used (SVM and LOF), we noticed that when using a good microphone, SVM had slightly better results, while with the poorer quality microphone LOF gave somewhat better results. The two algorithms that we used had no problems in distinguishing between identical devices – in our experiments, we used three identical Samsung J5's and the algorithms did not have any problems in distinguishing between them. We also tried to impersonate a device by recording its output with another device and replay the recording, but the system was resilient against such attacks.

Our investigations gave promising results but clearly, the practical adoption of such a system, especially in the automotive context, requires more investigations, which we leave as potential future work.

Author Contributions: Conceptualization, A.A., B.G.; methodology, A.A., P.I., C.B., B.G.; software, A.A., C.B., A.B. and P.I.; validation, A.A. and A.B.; investigation, A.A.; resources, A.B. and B.G.; writing—original draft preparation, A.A., A.B., P.I., C.B. and B.G.; writing—review and editing, A.A., A.B., P.I. and B.G.; supervision, B.G.; project administration, B.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lee, Y.; Li, J.; Kim, Y. MicPrint: acoustic sensor fingerprinting for spoof-resistant mobile device authentication. In Proceedings of the Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, 2019, pp. 248–257.
2. Das, A.; Borisov, N.; Caesar, M. Do you hear what I hear? Fingerprinting smart devices through embedded acoustic components. In Proceedings of the Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, 2014, pp. 441–452.
3. Car connectivity consortium. https://carconnectivity.org/. Accessed: 2025-10-4.
4. Grassi, P.A.; Garcia, M.E.; Fenton, J.L. Draft nist special publication 800-63-3 digital identity guidelines. *National Institute of Standards and Technology, Los Altos, CA* **2017**.
5. Jacomme, C.; Kremer, S. An extensive formal analysis of multi-factor authentication protocols. *ACM Transactions on Privacy and Security (TOPS)* **2021**, *24*, 1–34.
6. Two-factor authentication for Apple ID. https://support.apple.com/en-us/HT204915. Accessed: 2023-04-11.
7. The safer way to sign in to all of your online accounts. https://safety.google/authentication/. Accessed: 2023-04-11.
8. How two-factor authentication works on Facebook. https://www.facebook.com/help/148233965247823. Accessed: 2023-04-11.
9. Multifactor authentication in Azure AD. https://www.microsoft.com/en-us/security/business/identity-access/azure-active-directory-mfa-multi-factor-authentication. Accessed: 2023-04-11.
10. Google Authenticator Turn on 2-Step Verification. https://googleauthenticator.net/. Accessed: 2023-04-11.
11. Fauzi, M.A.; Yang, B. Audiouth: Multi-factor authentication based on audio signal. In Proceedings of the Proceedings of the Future Technologies Conference (FTC) 2020, Volume 3. Springer, 2021, pp. 935–946.
12. Shrestha, P.; Truong, H.; Toivonen, P.; Saxena, N.; Tarkoma, S.; Nurmi, P. Chirp-Loc: Multi-factor authentication via acoustically-generated location signatures. *Pervasive and Mobile Computing* **2022**, *88*, 101720.
13. Luo, J.N.; Tsai, M.H.; Lo, N.W.; Kao, C.Y.; Yang, M.H. Ambient audio authentication. *Mathematical Biosciences and Engineering* **2019**, *16*, 6562–6586.
14. Wang, M.; Yan, S.; Wang, W.; Jing, J. Secure zero-effort two-factor authentication based on time-frequency audio analysis. *International Journal of Information and Computer Security* **2022**, *18*, 237–261.
15. Nassi, B.; Pirutin, Y.; Shamir, A.; Elovici, Y.; Zadov, B. Lamphone: Real-time passive sound recovery from light bulb vibrations. *Cryptology ePrint Archive* **2020**.
16. Liao, Q.; Huang, Y.; Huang, Y.; Wu, K. An eavesdropping system based on magnetic side-channel signals leaked by speakers. *ACM Transactions on Sensor Networks* **2024**, *20*, 1–30.
17. Li, W.; Spolaor, R.; Luo, C.; Sun, Y.; Chen, H.; Zhang, G.; Yang, Y.; Cheng, X.; Hu, P. Acoustic Eavesdropping from Sound-Induced Vibrations with Multi-Antenna mmWave Radar. *IEEE Transactions on Mobile Computing* **2025**.
18. Shao, Z.; Su, Y.; Du, Y.; Huang, S.; Yang, T.; Liu, H.; Ren, Y.; Liu, B.; Dai, H.; Li, S. OISMic: Acoustic Eavesdropping Exploiting Sound-induced OIS Vibrations in Smartphones. In Proceedings of the 2024 21st Annual IEEE International Conference on Sensing, Communication, and Networking (SECON). IEEE, 2024, pp. 1–9.
19. pickle — Python object serialization. https://docs.python.org/3/library/pickle.html. Accessed: 2023-04-16.
20. Berdich, A.; Groza, B.; Mayrhofer, R.; Levy, E.; Shabtai, A.; Elovici, Y. Sweep-to-Unlock: Fingerprinting Smartphones based on Loudspeaker Roll-off Characteristics. *IEEE Transactions on Mobile Computing* **2021**.
21. Zhou, Z.; Diao, W.; Liu, X.; Zhang, K. Acoustic fingerprinting revisited: Generate stable device id stealthily with inaudible sound. In Proceedings of the Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, 2014, pp. 429–440.
22. Buchholz, R.; Kraetzer, C.; Dittmann, J. Microphone classification using Fourier coefficients. In Proceedings of the International Workshop on Information Hiding. Springer, 2009, pp. 235–246.
23. Ikram, S.; Malik, H. Microphone identification using higher-order statistics. In Proceedings of the Audio engineering society conference: 46th international conference: audio forensics. Audio Engineering Society, 2012.
24. Bojinov, H.; Michalevsky, Y.; Nakibly, G.; Boneh, D. Mobile device identification via sensor fingerprinting. *arXiv preprint arXiv:1408.1416* **2014**.

25. Hafeez, A.; Malik, K.M.; Malik, H. Exploiting frequency response for the identification of microphone using artificial neural networks. In Proceedings of the Audio Engineering Society Conference: 2019 AES International Conference on Audio Forensics. Audio Engineering Society, 2019.
26. Hanilci, C.; Ertas, F.; Ertas, T.; Eskidere, Ö. Recognition of brand and models of cell-phones from recorded speech signals. *IEEE Transactions on Information Forensics and Security* **2011**, *7*, 625–634.
27. Eskidere, Ö. Source microphone identification from speech recordings based on a Gaussian mixture model. *Turkish Journal of Electrical Engineering and Computer Sciences* **2014**, *22*, 754–767.
28. Aggarwal, R.; Singh, S.; Roul, A.K.; Khanna, N. Cellphone identification using noise estimates from recorded audio. In Proceedings of the 2014 International Conference on Communication and Signal Processing. IEEE, 2014, pp. 1218–1222.
29. Hanilçi, C.; Kinnunen, T. Source cell-phone recognition from recorded speech using non-speech segments. *Digital Signal Processing* **2014**, *35*, 75–85.
30. Qin, T.; Wang, R.; Yan, D.; Lin, L. Source cell-phone identification in the presence of additive noise from CQT domain. *Information* **2018**, *9*, 205.
31. Baldini, G.; Amerini, I. Smartphones identification through the built-in microphones with convolutional neural network. *IEEE Access* **2019**, *7*, 158685–158696.
32. Berdich, A.; Groza, B.; Levy, E.; Shabtai, A.; Elovici, Y.; Mayrhofer, R. Fingerprinting Smartphones Based on Microphone Characteristics From Environment Affected Recordings. *IEEE Access* **2022**, *10*, 122399–122413.
33. Jin, C.; Wang, R.; Yan, D. Source smartphone identification by exploiting encoding characteristics of recorded speech. *Digital Investigation* **2019**, *29*, 129–146.
34. Kotropoulos, C.; Samaras, S. Mobile phone identification using recorded speech signals. In Proceedings of the 2014 19th International Conference on Digital Signal Processing. IEEE, 2014, pp. 586–591.
35. Hon, T.K.; Wang, L.; Reiss, J.D.; Cavallaro, A. Audio fingerprinting for multi-device self-localization. *IEEE/ACM Transactions on Audio, Speech, and language processing* **2015**, *23*, 1623–1636.
36. Pillos, A.; Alghamidi, K.; Alzamel, N.; Pavlov, V.; Machanavajhala, S. A Real-Time Environmental Sound Recognition System for the Android OS. In Proceedings of the DCASE, 2016, pp. 75–79.
37. Ba, Z.; Piao, S.; Ren, K. Defending against Speaker Fingerprinting Based Device Tracking for Smartphones. In Proceedings of the 2017 IEEE Symposium on Privacy-Aware Computing (PAC). IEEE, 2017, pp. 188–189.
38. Júnior, P.R.M.; Bondi, L.; Bestagini, P.; Tubaro, S.; Rocha, A. An in-depth study on open-set camera model identification. *IEEE Access* **2019**, *7*, 180713–180726.
39. Kim, Y.; Lee, Y. CamPUF: physically unclonable function based on CMOS image sensor fixed pattern noise. In Proceedings of the Proceedings of the 55th annual design automation conference, 2018, pp. 1–6.
40. Cozzolino, D.; Marra, F.; Gragnaniello, D.; Poggi, G.; Verdoliva, L. Combining PRNU and noiseprint for robust and efficient device source identification. *EURASIP Journal on Information Security* **2020**, *2020*, 1–12.
41. Van Goethem, T.; Scheepers, W.; Preuveneers, D.; Joosen, W. Accelerometer-based device fingerprinting for multi-factor mobile authentication. In Proceedings of the International Symposium on Engineering Secure Software and Systems. Springer, 2016, pp. 106–121.
42. Ding, Z.; Ming, M. Accelerometer-based mobile device identification system for the realistic environment. *IEEE Access* **2019**, *7*, 131435–131447.
43. Hupperich, T.; Hosseini, H.; Holz, T. Leveraging sensor fingerprinting for mobile device authentication. In Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, 2016, pp. 377–396.
44. Das, A.; Borisov, N.; Chou, E. Every Move You Make: Exploring Practical Issues in Smartphone Motion Sensor Fingerprinting and Countermeasures. *Proc. Priv. Enhancing Technol.* **2018**, *2018*, 88–108.
45. Das, A.; Borisov, N.; Caesar, M. Tracking Mobile Web Users Through Motion Sensors: Attacks and Defenses. In Proceedings of the NDSS, 2016.
46. Stöber, T.; Frank, M.; Schmitt, J.; Martinovic, I. Who do you sync you are? smartphone fingerprinting via application behaviour. In Proceedings of the Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks, 2013, pp. 7–12.
47. Khodzhaev, Z.; Ayyildiz, C.; Kurt, G.K. Device Fingerprinting for Authentication. In Proceedings of the National Conference on Electrical and Electronics Engineering (ELECO), Bursa, Turkey, 2018.
48. Cheng, Y.; Ji, X.; Zhang, J.; Xu, W.; Chen, Y.C. Demicpu: Device fingerprinting with magnetic signals radiated by cpu. In Proceedings of the Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 1149–1170.
49. JTransforms library. https://github.com/wendykierp/JTransforms?tab=readme-ov-file. Accessed: 2025-10-4.
50. REW. https://www.roomeqwizard.com. Accessed: 2025-10-4.
51. Heroku website. https://www.heroku.com/home. Accessed: 2025-10-4.

52. Volley website. https://google.github.io/volley/. Accessed: 2025-10-4.
53. Flask website. https://flask.palletsprojects.com/en/2.2.x/. Accessed: 2025-10-4.
54. OC-SVM website. https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html. Accessed: 2025-10-4.
55. Cortes, C.; Vapnik, V. Support-vector networks. *Machine learning* **1995**, *20*, 273–297.
56. LOF algorithm website. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html. Accessed: 2025-10-4.
57. Breunig, M.M.; Kriegel, H.P.; Ng, R.T.; Sander, J. LOF: identifying density-based local outliers. In Proceedings of the Proceedings of the 2000 ACM SIGMOD international conference on Management of data, 2000, pp. 93–104.
58. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **2011**, *12*, 2825–2830.
59. Buitinck, L.; Louppe, G.; Blondel, M.; Pedregosa, F.; Mueller, A.; Grisel, O.; Niculae, V.; Prettenhofer, P.; Gramfort, A.; Grobler, J.; et al. API design for machine learning software: experiences from the scikit-learn project. In Proceedings of the ECML PKDD Workshop: Languages for Data Mining and Machine Learning, 2013, pp. 108–122.
60. Novelty and Outlier Detection. https://scikit-learn.org/stable/modules/outlier_detection. Accessed: 2023-04-11.
61. Aggarwal, C.C.; Hinneburg, A.; Keim, D.A. On the surprising behavior of distance metrics in high dimensional space. In Proceedings of the Database Theory—ICDT 2001: 8th International Conference London, UK, January 4–6, 2001 Proceedings 8. Springer, 2001, pp. 420–434.
62. Cai, W.; Chen, W.; Fang, J.; Holm, S. A survey on fractional derivative modeling of power-law frequency-dependent viscous dissipative and scattering attenuation in acoustic wave propagation. *Applied Mechanics Reviews* **2018**, *70*.
63. Grassi, P.A.; Garcia, M.E.; Fenton, J.L. Digital Identity Guidelines – Authentication and Lifecycle Management (SP 800-63B). National Institute of Standards and Technology (NIST), 2017. Special Publication 800-63B, Gaithersburg, MD, USA, https://doi.org/10.6028/NIST.SP.800-63b.
64. OWASP Foundation. OWASP Top 10 – 2021: The Ten Most Critical Web Application Security Risks. https://owasp.org/www-project-top-ten/, 2021. Accessed: 2025-10-04.
65. International Organization for Standardization.; International Electrotechnical Commission. ISO/IEC 27001:2022 – Information Security, Cybersecurity and Privacy Protection – Information Security Management Systems – Requirements. ISO Standard, 2022. ISO, Geneva, Switzerland.
66. International Organization for Standardization.; International Electrotechnical Commission. ISO/IEC 27002:2022 – Information Security, Cybersecurity and Privacy Protection – Information Security Controls. ISO Standard, 2022. ISO, Geneva, Switzerland.
67. Linux sysfs documentation. https://www.kernel.org/doc/Documentation/ABI/testing/sysfs-class-thermal. Accessed: 2025-10-4.
68. Lucertini, M.; Botti, T.; Sanjust, F.; Cerini, L.; Lucertini, L.; Sisto, R.; et al. High Altitude Performance of Loudspeakers and Potential Impact on Audiometric Findings. *Aerospace Medicine and Human Performance* **2019**, *90*, 655–659.
69. Google colaboratory. https://colab.google/. Accessed: 2025-10-4.