Listening between the Bits: Privacy Leaks in Audio Fingerprints

Moritz Pfister[1⋆], Robert Michael[1⋆], Max Boll[1], Cosima Körfer[1], Konrad Rieck[2,3,4], and Daniel Arp[2,3⋆⋆]

[1] Technische Universität Braunschweig
[2] Berlin Institute for the Foundations of Learning and Data (BIFOLD)
[3] Technische Universität Berlin

Abstract. Audio content recognition is an emerging technology that forms the basis for mobile services, such as automatic song recognition, second-screen synchronization, and broadcast monitoring. The technology utilizes *audio fingerprints*, short patterns that are extracted from audio recordings of a smartphone and enable the identification of specific content. These fingerprints are generally considered privacy-friendly, as they contain minimal information of the original signal. As a result, mobile applications have emerged in the past few years that silently monitor user habits by collecting such audio fingerprints in the background. In this paper, we systematically examine whether audio fingerprints leak sensitive information from the recording environment and potentially violate the privacy of smartphone users. To this end, we analyze three popular audio recognition solutions and develop attacks to infer sensitive information from their fingerprints. To the best of our knowledge, we are the first to show that the identification of speakers and words in the fingerprints is possible. Based on our analysis, we conclude that current audio fingerprints do not sufficiently protect privacy and should be used with great caution.

Keywords: audio content recognition · audio fingerprints · privacy · machine learning · mobile security

1 Introduction

Audio content recognition (ACR) is a recent technology for identifying content in audio recordings. In contrast to the direct analysis of audio signals, ACR relies on so-called *audio fingerprints* [47,40,37]. These small patterns are extracted from an audio signal and characterize its content in compressed form. Audio fingerprints can be efficiently transmitted and matched against large databases, enabling the identification of different types of audio content. While ACR has been initially developed for recognizing music [48,31], it is also used for tracking

⋆ Authors contributed equally to this work.
⋆⋆ Corresponding author: d.arp@tu-berlin.de

the behavior of users via their smartphones. For instance, mobile frameworks, such as Zapr [53] and Alphonso [25], monitor the acoustic environment of the smartphone in regular intervals to track the user's broadcast media consumption. These frameworks are not mere proof-of-concepts, but have been discovered in popular applications with millions of downloads [30,44].

A premise underlying this use of ACR is that audio fingerprints are privacy-friendly. That is, by condensing a recording into patterns of a few bytes, most of the information of the input signal is lost. Fingerprints are thus considered as "hashes" that can only be matched against other fingerprints but provide no further clues on the signal. Some providers of ACR thus even claim the collected data to be non-personal and boast about their privacy-friendly services [52].

Technically, fingerprinting algorithms compress an audio signal by identifying prominent frequency peaks and aggregating them in a fixed-size format. This compression destroys any continuous signal that could directly leak information. However, when the fingerprints are generated in a personal environment, the extracted peaks may capture nearby voices and sounds. While the resulting fingerprints do not match any known patterns in the databases of ACR services, it is unclear whether they contain sensitive information about speakers present in the environment that can be potentially misused by adversaries.

To the best of our knowledge, this is the first paper to investigate the privacy risks of audio fingerprinting systematically. In particular, we analyze three popular ACR solutions and their underlying fingerprint generation. Based on this analysis, we develop three attacks for inferring sensitive information from the raw bits of the fingerprints. These attacks build on machine learning to model temporal relations between frequency peaks and uncover different types of information. They enable us to (a) differentiate between speech and music signals, (b) identify the voices of individual speakers, and (c) detect a fixed set of spoken words within the fingerprints. In our evaluation, these attacks allow identifying 40 speakers with up to 90% accuracy and 10 words (digits) with up to 82% accuracy, demonstrating that the generated audio fingerprints still contain sensitive information.

While our attacks cannot undo the strong compression of audio fingerprints, they serve as a proof of concept for the reconstruction of personal information. We conclude that current ACR techniques do not adequately protect privacy and must not be used for sensitive tasks, such as monitoring user habits.

In summary, we make the following contributions:

- *Analysis of ACR solutions.* We provide the first analysis of currently available ACR solutions and the underlying fingerprinting mechanisms.
- *New attacks on audio fingerprints.* We propose three attacks that infer sensitive information from audio fingerprints of popular ACR solutions.
- *Code release.* To foster further research in this direction, we publish the source code and data used in our experiments[4].

The remainder of this paper is organized as follows: We provide an overview of related work in Section 2 and ACR solutions including the underlying fingerprint

[4] https://github.com/acr-privacy/attacks

generation in Section 3. Our attacks are introduced in Section 4 and empirically evaluated in Section 5. We discuss countermeasures and limitations in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

There exists various research related to this paper, which can be broadly divided into three different groups. We briefly discuss them in the following.

The first group of papers revolves around audio content recognition. In particular, early works already allow a reliable, noise-robust re-identification of audio content [15,47]. These are the basis for further improvements like increasing the robustness towards noise [20,33,39], and dealing with tempo and frequency shifts [40]. Although ACR algorithms are widely used, only few works exist that assess the potential privacy and security implications. On the security side, several works discuss evasion attacks challenging the robustness of audio fingerprinting [45,37]. On the privacy side, Knospe et al. [21] propose a privacy-enhanced fingerprinting algorithm that increases the entropy of extracted patterns to hinder attacks. Similarly, Celosi et al. [5] observe that Apple's digital voice assistant Siri is prone to dictionary attacks, allowing an adversary to identify pre-defined voice commands. Another large body of research demonstrates unintentional data leaks, e.g., methods for inferring sensitive information from encrypted network and telephone communication [50,26,36], inferring the TV content by analyzing the flickering lights emitted by TVs [51], or de-pseudomizing households by analysing pseudomized consumption data transmitted by smart meters [18]. More recently, various works have found further sources for data leakage in mobile and smart home environments [19,35]. Finally, Schlegel et al. [38] explore a mobile malware that stealthily extracts sensitive information through audio analysis.

Finally, there is a large amount of literature dealing with mobile device tracking. We discuss that various companies (e.g., [53,44]) developed techniques to profile the media consumption habits of smartphone users with audio fingerprints. Arp et al. [2] and Mavroudis et al. [29] reveal efforts where companies use ultrasonic beacons for similar purposes. The examination of potential privacy risks of tracking smartphone users via Bluetooth [14,23] demonstrates the possibility of cross-app tracking via fingerprints derived from BLE packets. Other works focus on privacy issues of cross-device tracking in detail [54,4] and mobile device tracking in general [6,35].

3 Audio Content Recognition

We start with an overview of the audio fingerprinting solutions considered in this paper and how they are used in practice (Section 3.1). Furthermore, we describe the technical principles, as these are essential to understand the attacks presented later in this paper (Section 3.2).

3.1 ACR Solutions

For our analysis, we consider three popular solutions for audio content recognition that focus on different application scenarios.

Quad-based Fingerprinting (QBF). The first approach we investigate is an academic method proposed by Sonnleitner and Widmer [40]. We refer to this method as *quad-based audio fingerprinting (QBF)*. We select this approach, as it is an extended variant of the classic method by Wang [47], which forms the foundation for many ACR solutions—including the popular Shazam app. Since the publication by Wang [47] lacks implementation details, we use QBF as a replacement, where we have concrete knowledge of its internal parameters and other essential information that allows us to fully re-implement the method.

Zapr. As the second approach, we consider the ACR solution from the company Zapr [22]. Zapr used to be a popular commercial ACR solution that was active until mid 2022. The company focused exclusively on media consumption monitoring and targeted advertising [53]. According to the company's website, their solution was present on more than 200 million mobile devices, backed by several large global media companies [53]. Although the company closed down, its technology can still serve as a prototypical example of how audio fingerprints are used for user tracking.

Zapr offered a software development kit (SDK) for media consumption monitoring to their customers. Their SDK included a dedicated recording service that kept running in the background and enabled apps to continuously monitor the acoustic environment—even if the apps were not active. To achieve this, the respective apps notified their users and requested permissions to access the location and the microphone of the device. Like companies with similar business models [44,24], Zapr claimed that it is impossible to reverse-engineer these audio fingerprints and gain insights into the recorded content [52]. Consequently, the privacy of users depended entirely on whether and what information are recoverable from the fingerprints submitted to Zapr.

ACRCloud. Finally, we consider the commercial ACR solution of the company ACRCloud. Among others, their customers include Deezer [8], MusixMatch [31], and Huawei. Together, these companies reach a global customer base with several millions of users, making ACRCloud one of the major solutions on the market. Besides music recognition for MusixMatch and Deezer, ACRCloud also focuses on custom content recognition, for example, in the segments of broadcast monitoring and advertising [1]. While apps like MusixMatch and Deezer require their users to start the audio recording explicitly, it is unknown whether this technology is also embedded in apps with a business model similar to Zapr. Moreover, due to the closed-source nature of ACRCloud, technical details of the fingerprinting algorithm are not publicly available. We address this problem when deriving our attacks in Section 4 using techniques for code instrumentation.

Further Approaches. Although there exist further approaches for employing ACR in the wild, we focus on the presented solutions, as they are prototypical for how audio fingerprints are used in practice, including foreground (ACRCloud) and background recording (Zapr).

3.2 Audio Fingerprinting

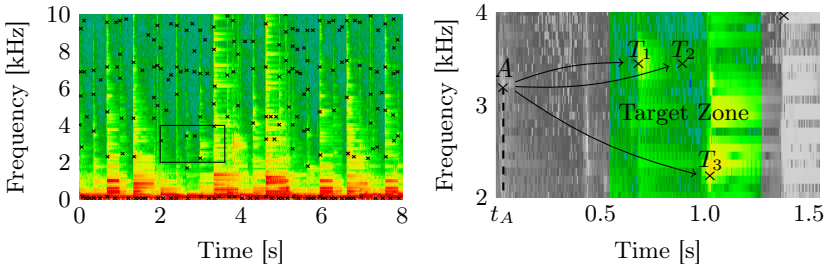
Before we introduce our attacks against ACR solutions, let us first briefly review the basic requirements and concepts of audio fingerprinting. The general pipeline that all ACR methods share can be broadly divided into three steps: *(a) pre-processing*, *(b) fingerprint extraction*, and *(c) fingerprint matching*. All solutions share various requirements they need to fulfill. We first discuss these requirements, and then explain the different steps of the generation pipeline. Here, we mainly focus on the method by Wang [47], because it has inspired several other recent approaches (e.g., [40,22]).

Requirements. In order to enable successful fingerprinting in different application scenarios, ACR solutions need to realize a trade-off between four requirements, which influence the characteristics of the underlying algorithms.

- *Reliable identification.* An ACR solution needs to reliably identify audio content solely based on a short recording. Moreover, it should produce only very few false-positive results during practical operation.
- *Noise robustness.* The solution needs to cope with strong noise. For instance, it should allow detecting media content even in locations with high background noise, such as in bars or on concerts.
- *High compression.* To enable efficient transfer and matching of fingerprints, they should only contain the necessary information for re-identifying a signal and compress the underlying recording significantly.
- *Computational efficiency.* Finally, the fingerprints need to be generated on devices with limited computational and memory resources, such as smartphones. Thus, the generation process needs to be very efficient.

Pre-processing. As the first step, the audio signal is prepared for extracting a fingerprint. Since not all frequency components are necessary for audio content recognition, the original signal is usually downsampled before a fingerprint is created (e.g., [15,47]). Audio recordings on smartphones are commonly generated as a stereo signal sampled with a frequency of 44.1 kHz. Hence, ACR solutions first merge the two channels into a mono track and then filter high frequencies using a simple low-pass filter.

Fingerprint Extraction. To access relevant features of an audio signal, most ACR methods map the underlying time series to the frequency domain and compute the spectrum of the signal. That is, they represent the signal as a mixture of frequency components using standard algorithms, such as the short-time Fourier transform (STFT). This frequency spectrum allows analyzing the changes of the



(a) (b)

Fig. 1: (a) Spectrogram of an eight-second-long audio signal. Prominent frequency peaks are indicated by black markers. (b) Derivation of subfingerprints by combining the anchor point A (left) with peaks in the respective target zone.

signal's frequency components over time. As an example, Figure 1a depicts the spectrogram of an audio signal that exhibits prominent frequency changes.

While most audio fingerprinting methods rely on a spectral analysis, the derived spectra usually differ in their time and frequency resolution. However, despite these differences, the algorithms proceed similarly and divide the frequency range in various non-overlapping frequency bands, also called *frequency bins*.

After calculating the spectrum of a signal, audio fingerprints are derived from prominent frequency components. The exact method used to create these fingerprints may differ, but they all have in common that they analyse the spectrum for characteristic peaks and their temporal relation. The extracted peaks form the basis for assembling the audio fingerprint. To give a concrete example of the fingerprint generation process, let us consider the approaches by Wang [47] and Sonnleitner et al. [40]. These methods first identify peaks in the spectrum of the signal. Afterward, they select so-called *anchor points* from these peaks and assign a *target zone* following each anchor point.

The target zone is a small area of the spectrum in a certain temporal distance to the corresponding anchor point A. Frequency peaks T_j in this zone are called target points (see Figure 1b) and capture short temporal relations between frequencies in the signal. For each anchor point A, Wang then constructs multiple *subfingerprints* by combining A with a selected number of peaks T_j in the target zone. In particular, each resulting subfingerprint is a tuple $f_i = (b_A, b_T, \Delta t)$, where b_A denotes the frequency bin of the anchor point, b_T the frequency bin of the considered target point and $\Delta t = t_T - t_A$ their temporal distance.

In the method of Wang, a subfingerprint f_i comprises 30 bits, encoding each frequency bin and the time difference with 10 bits each. In contrast, Sonnleitner et al. extract 32-byte-long subfingerprints. Similarly, other parameters, such as the target zone width, differ depending on the approach. To give an intuition, Sonnleitner et al. [40] use each extracted peak once as an anchor point and consider a 0.8 seconds long window as target zone, which is 0.9 seconds away from the current anchor point. Despite these configuration details, most audio

fingerprinting methods rely on characterizing the signal through subfingerprints that capture frequency pairs and their temporal distance.

Note that the extracted subfingerprints alone are not sufficient to identify a signal, as similar frequency combinations may occur in different audio content. Therefore, multiple subfingerprints are required together with the time coordinate of their respective anchor point t_A to allow for a reliable identification. The final audio fingerprint of a signal s is typically described as a sequence $f(s) = ((f_1, t_{A_1}), \ldots, (f_N, t_{A_N}))$. There exist fingerprinting methods where the number of extracted subfingerprints N is fixed (e.g., [15]), while for other approaches N depends on the audio signal to be fingerprinted (e.g., [47,40]). As we discuss in Section 4, our attack method partially leverages this structural information.

On the server side, the generated fingerprints are usually stored in a database along with some metadata of the original audio signal, such as a track name. Finally, the fingerprints in the database can be used to match incoming fingerprints derived from an audio signal recorded by a client app.

Fingerprint Matching. Once the database has been constructed successfully, one can use it to re-identify audio content solely based on small snippets of the audio signal. In particular, a client application records a short audio snippet of only a few seconds, and extracts a fingerprint from it as described before. Then, the app transmits the fingerprint to the server. On the server side, the fingerprint can be matched against the fingerprints stored in the database.

4 Attacks against Audio Fingerprints

We now introduce our attacks against ACR. To this end, we first define the threat model underlying our attacks (Section 4.1), then discuss possible attack scenarios (Section 4.2), and finally present the details of our attack method (Section 4.3).

4.1 Threat Model

In our threat model, we assume that an attacker has access to audio fingerprints from an ACR solution. The attacker could be directly involved in the service that provides the solution or a third party that has access to the matched fingerprints. Numerous ways in which third parties could access these fingerprints are conceivable. These include data trading, service outsourcing, a false sense of privacy, law enforcement, or unintentionally through a data breach.

Furthermore, we assume that at least some of the collected audio fingerprints contain sensitive information the attacker is interested in. The attacker's goal is to reconstruct this information from the attacked fingerprints, such as speaker identities, spoken words, or further attributes of the acoustic environment at the time of recording. This implies that not only the privacy of the device owner is threatened, but even the privacy of all persons present during the fingerprinting process. As a result, our analysis is based on the following assumptions regarding the attacker's knowledge and capabilities:

- *Audio fingerprints.* The attacker has access to audio fingerprints from the device of a victim, for example, when they are transferred to the ACR service. However, the attacker neither needs to know the original signal nor a reverse mapping that allows her to reconstruct it. We refer to the collected audio fingerprints as *attacked (audio) fingerprints*.
- *Black-box access.* The attacker has at least black-box access to the fingerprinting algorithm that produces the attacked audio fingerprints. This capability enables them to generate new fingerprints at their will, for instance, by instrumenting the code of a mobile app.
- *Auxiliary data.* The attacker has access to auxiliary data related to the sensitive information of interest. Here, publicly available data can already be sufficient to run successful attacks. For instance, the attacker might use audio or video content published on social media platforms to target specific persons of interest in the attacked fingerprints.

We argue that such an attacker is realistic. First of all, there exist various solutions that generate audio fingerprints and send them to external servers [44,53]. Malicious employees or dubious companies might actively analyze the collected data to derive sensitive information from these audio fingerprints. Thus, users cannot always trust companies to handle their data responsibly. However, even if we assume that all companies are trustworthy, history has repeatedly shown that this does not eliminate the risk of data leaks to malicious actors [9,43]. For instance, Deezer, a company that uses ACRCloud for content recognition (see Appendix A), has been reported as a recent victim of data theft [9].

Secondly, it is a realistic assumption that the attacker has at least black-box access to the fingerprinting algorithm. In this paper, we demonstrate that a determined attacker does not need to know the exact implementation of the ACR algorithm in order to conduct the attack. Instead, they can instrument code in mobile apps, and generate fingerprints from custom data. More dedicated attackers could also use common reverse engineering methods to gain insights on the internals of specific methods, and use them to improve possible attacks. In the next section, we discuss in more detail how an attacker can generate fingerprints despite having only black-box access to the targeted ACR solution.

Finally, there are several public datasets for audio recognition that can be used as auxiliary data by an attacker. These datasets can be combined with attack-specific data, such as audio samples of a specific speaker, to provide the necessary foundation for the attack. The adversary might collect this data, for instance, from social media platforms, such as YouTube or Instagram, where many users upload personal media containing usable audio samples.

4.2 Attack Scenarios

The privacy impact of our attacks depends on the type of leaked information. Therefore, we define three attack scenarios, each focusing on a different information type. In these scenarios, we assume that the attacker has collected matching auxiliary data and now aims at inferring the corresponding information from captured audio fingerprints.

- *Distinguishing speech.* In the first scenario, the adversary wants to distinguish fingerprints derived from speech from those containing other signals. Since audio content recognition is often used to identify music tracks, we focus on distinguishing fingerprints derived from human speech and music, respectively. As auxiliary data, the attacker has access to a collection of corresponding audio recordings.
- *Identifying speakers.* In the second scenario, the attacker is interested in identifying the speaker of a voice signal that is captured by an audio fingerprint. This identification allows them to associate the recorded signal with a specific person, thus potentially compromising their privacy. As auxiliary data, we assume a dataset containing audio recordings of the targeted speaker and other voices.
- *Recognizing words.* Finally, the attacker tries to reconstruct parts of the spoken content from the fingerprints. If successful, the attacker can retrieve different sensitive information. We restrict our attack to identifying the English digits zero to nine, which already poses a privacy risk when, for example, a credit card or phone number is spelled. As auxiliary data, we assume that a dataset with spoken digits is available to the attacker.

Although we examine each of these attack scenarios individually, they can be chained together to generate a more powerful attack. Further, it is possible to add other recognition tasks, such as identifying ambient sounds or other spoken words, to strengthen our attack. We thus consider the three scenarios as prototypical for a group of recognition tasks that are applicable to audio fingerprints.

4.3 Attack Method

As the captured fingerprints contain only a fraction of the information of the original signal, the attacker needs to carefully engineer an attack method to reconstruct the information they are interested in. This method comprises three steps, which we discuss in the following.

Fingerprint Generation. As the first step, the attacker needs to generate a representative set of fingerprints based on the auxiliary data (see Section 4.1). This set serves as training data and enables machine-learning methods to infer a recognition model from the contained frequency peaks and their temporal relation. This fingerprint generation, however, depends on whether the adversary has white-box or black-box access to the underlying algorithms.

From the three considered ACR solutions, we only have full access to the QBF method and thus simply implement the fingerprinting algorithm described in the original publication [40]. We only make slight adaption to the algorithm's parameters, such that it can handle short audio snippets of less than a second, which the original configuration cannot. Unfortunately, we cannot simply implement the audio fingerprinting algorithms of the other two solutions, as they are closed-source software.

To tackle this problem, we reverse engineer the relevant parts of the SDKs of the two solutions. We first perform a static analysis of popular apps[5][6] containing the SDKs. This is done to identify the respective methods for generating audio fingerprints along with the used parameters. Using the determined parameters, we use the analysis toolkit Frida [34] to dynamically instrument the apps' code and generate fingerprints within the apps, enabling us to obtain the exact same fingerprints as sent to the companies' backends. That is, we provide the methods prepared audio buffers and intercept the returned audio fingerprint data.

For Zapr, we find that it implements five different fingerprinting methods. As the manual effort to reverse engineer all of them is too high, we decide to select two of them for further examination. In the following, we refer to them as Zapr Alg1 and Zapr Alg2. As the first algorithm, we select Zapr Alg1, since its reverse-engineered interface exhibits similarities to the only publicly available description provided by the company [22]. Moreover, we select Zapr Alg2, which appears to be the default algorithm at the time of our analysis.

Model Architecture. Once representative fingerprints have been generated, and the corresponding byte sequences are available, we can finally use machine learning for inferring recognition models. Due to the sequential nature of our dataset we base our models on the transformer architecture [46]. Since we want to solve a classification task, we only utilize the encoder part of the transformer architecture, which has shown promising results in other problem domains (e.g., [11,13]).

In detail, the model receives fingerprints as input. Each input fingerprint f is a sequence of subfingerprints $f_1, \ldots, f_k$, where each subfingerprint is l bits long. The sequence length k and the number of bits l are specific to the concrete ACR solution or classification task. Fingerprints containing fewer than k subfingerprints are padded with all-zero subfingerprints until they reach the sequence length of k. First, each subfingerprint is mapped to a d-dimensional embedding vector, which also encodes information about the subfingerprint's position within the sequence. The embedded subfingerprints are then processed by successive encoder blocks that allow capturing the context within a sequence of audio fingerprints. After the encoder blocks, the sequential output is condensed using the attention-based pooling mechanism *SeqPool* [16], and then mapped to an n-dimensional vector. The resulting vector contains the logits for the corresponding classification task, where n is the number of classes (e.g., the number of words or speakers).

Model Training. We peform a hyperparameter selection and train a distinct model for each of the attack scenarios and each ACR solution. All models follow the architecture described above. However, we vary the embedding dimension d, the number of encoder blocks, and the number of attention heads. Throughout the experiments, we use a fixed number of bits l for most of the considered ACR solutions. Only for Zapr Alg2, we vary this parameter, as the exact length of the subfingerprints is unknown for this solution. An overview of the selected hyperparameters can be found in Table 2 in the Appendix.

[5] sha1: `3c0770204a5d769c1a22a4acb7f9d6a4dd12e55c`
[6] sha1: `5de8eb4098d2e35a2c3951a169bf9e19a680e2d4`

We train each model for 300 epochs using the weighted Adam optimizer, combined with a learning rate schedule based on cosine decay [27,28]. To avoid overfitting of our model, we apply three common regularization methods. In particular, we combine dropout [41], stochastic depth regularization [17], and label smoothing [42] during training.

Note that we make our code and the used datasets publicly available[7] to allow other researchers to reproduce the results, and access details on our feature extraction and model selection procedure.

5 Evaluation

We proceed to investigate the impact of the proposed attacks on the privacy of smartphone users. Following the three attack scenarios, we empirically evaluate whether an adversary can identify human speech, particular speakers, or spoken words in the content of the collected audio fingerprints.

5.1 Speech vs. Music

To run a successful attack, an attacker first needs to be able to filter out potentially interesting audio fingerprints that contain human speech. That is, the attacker wants to solve a binary classification problem, in which the learning model receives an audio fingerprint as input and predicts one of two possible classes.

Dataset. For this experiment, we combine audio samples of human speech from the popular LibriSpeech dataset [32] with music examples taken from the free music archive (FMA) dataset [10]. In particular, we randomly select the same number of samples from each of the datasets and split them into audio recordings with a duration of 3 seconds each. This is done to ensure that all audio samples in our dataset have a fixed length, so that the machine learning model cannot implicitly use the length of a sample as a shortcut for its decision [3]. We then employ the audio fingerprinting algorithms of the selected ACR solutions on these recordings. As all algorithms expect audio signals sampled at 8,000 Hz, we resample the files in the dataset accordingly. Our final dataset comprises a total of 61,624 audio recordings and respective fingerprints.

Experimental Setup. To avoid overfitting the neural network, we split the data into separate training, validation, and test partitions. That is, we use 35,512 samples for training, 10,112 samples for validation, and 16,000 samples for testing. Note that we ensure that no audio recording used for obtaining the samples in one partition is used in another. We then determine the best setup for our attack on the validation data, where we vary the hyperparameters (see Section 4.3). We finally measure the performance of the best setup on the test set.

[7] https://github.com/acr-privacy/attacks

Table 1: Overview of experimental results. For each setting, the table shows the Accuracy (Acc), Precision (P), Recall (R), and F1-Score (F1). The last row shows the results of a random guess by the attacker (RND).

	Speech				Speaker				Word			
	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1
QBF	.97	.97	.97	.97	**.90**	**.90**	**.90**	**.91**	.70	.70	.70	.70
ACR	**.98**	**.98**	**.98**	**.98**	**.90**	**.90**	**.90**	**.91**	**.82**	**.82**	**.82**	**.82**
ZA1	.95	.95	.95	.95	.64	.64	.64	.64	.58	.58	.58	.59
ZA2	.68	.68	.68	.68	.05	.03	.05	.03	.40	.40	.40	.40
RND	.50	.50	.50	.50	.03	.03	.03	.03	.10	.10	.10	.10

Results. Table 1 shows the results of this experiment. The attack achieves the highest success rates using the fingerprints of ACRCloud, with an accuracy of 98%. The results for QBF and Zapr Alg1 are only slightly worse, yielding reliable detection rates of 97% and 95%, respectively. While the attack is less effective against Zapr Alg2, we can yet classify 67% of the samples correctly—which still exceeds random guessing.

Overall, our results show that audio fingerprints contain enough information to reliably distinguish between music and speech signals. This result is not unexpected, as the attack scenario is close to the original purpose of ACR solutions. Nevertheless, our attack can serve as a preparatory step for further analysis and inference of other relevant information.

5.2 Speaker Identification

For the next experiment, we assume that the attacker was able to obtain fingerprints that solely contain human speech. They now want to identify the speakers that were recorded by the mobile device. To mimic this scenario, we design an experiment with n speakers. That is, the attacker tries to distinguish between n distinct classes from the audio fingerprints. In this setting, the final layer consists of n output neurons, where each neuron corresponds to one of the speakers.

Dataset. To evaluate the attack in this setting, we use the LibriSpeech dataset [32]. We randomly select audio recordings of 2×40 distinct speakers from the LibriSpeech dataset. Like in the previous experiments, we resample the audio files at 8,000 Hz and split them into chunks of three seconds, which are then used to generate the audio fingerprints. The resulting dataset contains 33,920 samples, with at least 400 instances per speaker.

Experimental Setup. We first split the LibriSpeech data into a calibration set and an evaluation set, each comprising 40 *different* speakers, and containing 16,720 and 17,200 audio samples, respectively. This split ensures that we do not overfit to peculiarities of some of the speakers. We then perform experiments on the

calibration set to select hyperparameters according to Section 4.3. Finally, we select the best attack setup and run the final experiment on the evaluation set.

Results. For QBF and ACRCloud, we receive a high accuracy of up to 90% (see Table 1). The attack success rate outperforms random guessing by large extent: As we deal with a 40-class problem, a random prediction only yields 2.5%. Only for Zapr Alg2, we obtain results close to random guessing. In contrast, however, we find that Zapr Alg1 is also prone to this attack, with an accuracy of 64%, proving that the attacked fingerprints still leak sensitive information.

In summary, we show with this experiment that audio fingerprints reveal enough information to identify individual speakers, which can be sensitive information in many cases. The attack fails on one algorithm only, i.e., Zapr Alg2. While at first glance this result suggests a higher robustness of Zapr Alg2, it should be interpreted with great caution due to our limited knowledge of the fingerprint structure. An attacker with greater knowledge of the internals of the fingerprinting algorithm might be able to also run a successful attack against these fingerprints as well.

5.3 Word Identification

In the third attack scenario, we also assume that the adversary has obtained audio fingerprints that contain human speech. However, unlike the previous experiment, the attacker is now interested in the spoken content stored in these fingerprints. To simulate this situation, we design an experiment where each audio fingerprint contains one of n spoken words. Here, we choose the English digits from zero to nine as the basis for these words, such that $n = 10$. The final layer of the neural network contains 10 neurons, where each output neuron represents one of the digits. While more complex attack scenarios are conceivable, even the identification of a limited set of words like digits poses a privacy problem, for example, when a PIN or credit card number is spelled out loud.

Dataset. For this experiment, we use the *Speech Commands* dataset [49] and extract all audio of spoken digits. The dataset already contains a prepared split into training and testing partitions. Therefore, we use this partitioning instead of generating our own. However, we prepare the training data by removing duplicates that would simplify the attack. Also, we balance the number of audio snippets per digit. Our final dataset includes a total of 36,462 audio samples, where 32,630 are used for training and 3,832 for testing.

Experimental Setup. We follow the same experimental procedure as described in Section 5.1, and select the setup that yields the best results on the validation data. The selected parameters are described in the Appendix (see Table 2). Finally, we evaluate its performance on the test set. In this case, however, the test set is not entirely balanced. Some digits occur slightly less frequently than others. To ensure a fair comparison, we macro-average the results to ensure that each digit is weighted equally.

Results. Table 1 shows the results of this experiment. In contrast to the identification of speakers, all considered solutions are vulnerable to this attack, although not to the same extent. In particular, we obtain the best results for ACRCloud with an accuracy of 82%, followed by QBF and Zapr Alg1 with 70% and 58%, respectively. To our surprise, the attack is also successful against Zapr Alg2 in 40% of the cases, where the identification of individual speakers failed. While the algorithm shows the highest robustness, the success of the attacker is still significantly higher than random guessing, which only yields 10% accuracy in this setting. Although our experiment builds on a simplified setup with a limited number of words, it is clear that spoken content can be identified in audio fingerprints, raising doubts about their privacy-friendliness. We suspect that similar attacks can be conducted with personal names, street addresses, and other sensitive terms that would compromise the privacy of smartphone users.

6 Discussion

Our experimental results in the previous sections show that audio fingerprints can indeed leak sensitive user information that might be misused by malicious parties. In the following, we sketch three possible countermeasures of increasing technical complexity to respond to this threat. Furthermore, we elaborate on limitations and possible biases in our experimental design.

Countermeasures. The first and simplest countermeasure is to notify users properly if the device is recording audio in the background. For instance, recent versions of mobile operating systems display an indicator on the device signaling an ongoing recording. While this approach should lower the risk that fingerprint generation takes place without the user's knowledge, it might be overlooked easily. An additional problem arises from the fact that most Android devices still run with Android 10 or lower and thus do not notify the users when the microphone is used in the background. Consequently, we argue that audio content recognition should only be performed at the user's request.

Secondly, cryptographic hashing can help to protect the sensitive data stored in the fingerprints. In particular, applying a cryptographic hash on the subfingerprints still allows matching them, while hindering the dissection and statistical analysis of the contained frequency peaks and temporal differences. However, to rule out dictionary attacks, the entropy of the fingerprint distribution has to be high enough. Further, a keyed hash function with a periodic key rotation should be used (see [21]). While the former remains questionable, the latter is ruled out by the business models in the ACR context.

Finally, we note the similarities between the technique of private set intersection (PSI) and the matching subfingerprints against a database. PSI enables two parties to generate an intersection of private sets without revealing the elements not in common. Several PSI approaches consider scenarios where a small set is intersected with a large set (e.g., [12,7]), thus reflecting the matching process in ACR solutions. Utilizing PSI, the service provider only learns of subfingerprints

present in her database, thus reducing the information available to the attacker. This results in a reduced attack effectiveness, as matching specific samples needs preparation in advance. PSI, however, incures a notable overhead. While this may be acceptable for applications with sporadic ACR usage, such as music recognition, this renders continuous tracking infeasible. As with cryptographic hashing, if the fingerprint distribution is of insufficient entropy, creating a dictionary and matching it against the incoming fingerprints remains possible.

Limitations. Although we have designed our experiments with great care, we could not show the leakage of sensitive data in all examined cases. Unfortunately, this does not automatically imply a proper protection of sensitive data by the ACR solutions in these cases. Instead, we faced several limitations that might have affected our experimental outcome, hindering us to extract the sensitive information reliably.

First of all, our experimental setting is not free from bias, as we perform our experiments under controlled conditions on mostly balanced datasets. However, as we intend to demonstrate possible privacy threats of ACR solutions in this paper, we argue that our setup is still appropriate for this purpose. Still, further research is required to better assess and grasp the exact impact of our findings and develop effective defenses against this privacy threat. On one hand, we have not systematically assessed the robustness of the attacks in the presence of background noise, which could potentially even serve as a simple defense mechanism. Overall, more advanced attacks against ACR fingerprints along with corresponding defenses should be explored in the future.

Secondly, we find that the examined ACR solutions are not vulnerable to the same extent. Most notably, we could not show that distinguishing between multiple speakers is possible when generating fingerprints with the Zapr Alg2 algorithm. However, we note that we have not been able to fully reverse engineer and understand the internals of this particular solution. Hence, even attacks on this algorithm might still be possible, given an adversary with enough background knowledge of the underlying algorithms.

Finally, in this paper, we consider the application of ACR in mobile apps only. However, as these solutions are part of a broad ecosystem that is difficult for externals to grasp, they are deployed in many different application scenarios as well [53,47,15,40,1]. A complete survey of this ecosystem is thus out of scope. Nonetheless, we think that our gained insights generalize to other ACR solutions, as we have selected two popular examples that are widespread and offer services and SDKs not limited to a specific app or use case. Still, further research is required to assess the security of other ACR methods as well.

7 Conclusion

In this work, we provide insights into the ACR ecosystem and demonstrate that the employed audio fingerprints reveal private information. The evaluation of our attacks gives evidence that audio fingerprints leak sensitive information and should not be used as a privacy protection mechanism.

Within the conducted experiments the proposed attacks identify sensitive information, achieving high accuracy values that are significantly better than random guessing. In particular, we can show that it is possible to identify speakers and spoken words in the audio fingerprints of three major ACR solutions using machine learning techniques.

These results raise serious privacy concerns, especially if this technology runs in the background without the users' knowledge. We conclude that these approaches are not privacy-friendly and caution should be exercised, especially when using them for background monitoring.

Acknowledgements. This work was funded by the German Federal Ministry of Education and Research (BMBF) under the grants BIFOLD24B and 16KIS1142K.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. ACRCloud: Acrcloud: Automatic content recognition services for doers. `https://www.acrcloud.com/` (2022), (last visited Apr. 22, 2024)
2. Arp, D., Quiring, E., Wressnegger, C., Rieck, K.: Privacy threats through ultrasonic side channels on mobile devices. In: Proc. of IEEE European Symposium on Security and Privacy (EuroS&P) (2017)
3. Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., Rieck, K.: Dos and don'ts of machine learning in computer security. In: Proc. of USENIX Security Symposium (2022)
4. Brookman, J., Rouge, P., Alva, A., Yeung, C.: Cross-device tracking: Measurement and disclosures. Proc. on Privacy Enhancing Technologies (PETS) **2017**(2) (2017)
5. Celosia, G., Cunche, M.: Discontinued privacy: Personal data leaks in apple bluetooth-low-energy continuity protocols. Proc. on Privacy Enhancing Technologies (PETS) **2020**(1) (2020)
6. Chatterjee, R., Doerfler, P., Orgad, H., Havron, S., Palmer, J., Freed, D., Levy, K., Dell, N., McCoy, D., Ristenpart, T.: The spyware used in intimate partner violence. In: Proc. of IEEE Symposium on Security and Privacy (S&P) (2018)
7. Chen, H., Laine, K., Rindal, P.: Fast private set intersection from homomorphic encryption. In: Proc. of ACM Conf. on Computer and Communications Security (CCS) (2017)
8. Deezer: Deezer | listen to music | online music streaming platform. `https://www.deezer.com` (2022), (last visited Apr. 22, 2024)
9. Deezer: Third party data breach – deezer support. `https://support.deezer.com/hc/en-gb/articles/7726141292317-Third-Party-Data-Breach` (2022), (last visited Apr. 22, 2024)
10. Defferrard, M., Benzi, K., Vandergheynst, P., Bresson, X.: FMA: A dataset for music analysis. In: Proc. of the Int. Society for Music Information Retrieval Conference (ISMIR) (2017)

11. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proc. of the Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT) (2019)
12. Dong, C., Chen, L., Wen, Z.: When private set intersection meets big data: an efficient and scalable protocol. In: Proc. of ACM Conf. on Computer and Communications Security (CCS) (2013)
13. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In: Proc. of Int. Conference on Learning Representations (ICLR) (2019)
14. Faragher, R., Harle, R.: Location fingerprinting with bluetooth low energy beacons. IEEE Journal on Selected Areas in Communications **33**(11), 2418–2428 (Nov 2015)
15. Haitsma, J., Kalker, T.: A highly robust audio fingerprinting system. In: Proc. of the Int. Society for Music Information Retrieval Conference (ISMIR) (2002)
16. Hassani, A., Walton, S., Shah, N., Abuduweili, A., Li, J., Shi, H.: Escaping the big data paradigm with compact transformers. CoRR **abs/2104.05704** (2021)
17. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep Networks with Stochastic Depth. In: Computer Vision – ECCV (2016)
18. Jawurek, M., Johns, M., Rieck, K.: Smart metering de-pseudonymization. In: Proc. of Annual Computer Security Applications Conference (ACSAC) (2011)
19. Kennedy, S., Li, H., Wang, C., Liu, H., Wang, B., Sun, W.: I can hear your alexa: Voice command fingerprinting on smart home speakers. In: Proc. of. IEEE Conference on Communications and Network Security (CNS) (2019)
20. Kim, H.G., Cho, H.S., Kim, J.Y.: Robust audio fingerprinting using peak-pair-based hash of non-repeating foreground audio in a real environment. Cluster Computing **19**(1) (2016)
21. Knospe, H.: Privacy-enhanced perceptual hashing of audio data. In: International Conference on Security and Cryptography (SECRYPT) (2013)
22. Konjeti, S., Potty, H., Kashyap, D.: Zapr audio fingerprinting. `https://www.music-ir.org/mirex/abstracts/2017/KP1.pdf` (2017)
23. Korolova, A., Sharma, V.: Cross-app tracking via nearby bluetooth low energy devices. In: Proc. of ACM Conference on Data and Applications Security and Privacy (CODASPY) (2018)
24. LG Ads Solutions: Alphonso ACR technology and consumer choice - lg ads[3]. `https://tinyurl.com/yp9t2dmz` (2018), (last visited Apr. 22, 2024)
25. LG Ads Solutions: Automatic content recognition. `https://alphonso.tv/` (2021), (last visited Apr. 22, 2024)
26. Liberatore, M., Levine, B.N.: Inferring the source of encrypted http connections. In: Proc. of ACM Conf. on Computer and Communications Security (CCS) (2006)
27. Loshchilov, I., Hutter, F.: SGDR: stochastic gradient descent with warm restarts. In: Proc. of Int. Conference on Learning Representations (ICLR) (2017)
28. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: Proc. of Int. Conference on Learning Representations (ICLR) (2019)
29. Mavroudis, V., Hao, S., Fratantonio, Y., Maggi, F., Kruegel, C., Vigna, G.: On the privacy and security of the ultrasound ecosystem. Proc. on Privacy Enhancing Technologies (PETS) **2017**(2) (2017)
30. Media, B.: Hotstar, newsdog and other indian apps are spying on your phone's mic. `https://beebom.com/hotstar-newsdog-apps-spying-phone-mic/` (2018), (last visited, Apr. 22, 2024)

31. Musixmatch S.p.A.: Musixmatch website. https://www.musixmatch.com (2022), (last visited Apr. 22, 2024)
32. Panayotov, V., Chen, G., Povey, D., Khudanpur, S.: Librispeech: An ASR corpus based on public domain audio books. In: IEEE Int. Conference on Acoustics, Speech and Signal Processing (ICASSP) (2015)
33. Park, M., Kim, H.R., Yang, S.H.: Frequency-temporal filtering for a robust audio fingerprinting scheme in real-noise environments. ETRI Journal **28**(4) (2006)
34. Ravnås, O.A.V.: Frida • a world-class dynamic instrumentation framework. https://frida.re/ (2022), last visited: Apr. 22, 2024
35. Reardon, J., Feal, Á., Wijesekera, P., On, A.E.B., Vallina-Rodriguez, N., Egelman, S.: 50 ways to leak your data: An exploration of apps' circumvention of the android permissions system. In: Proc. of USENIX Security Symposium (2019)
36. Rimmer, V., Preuveneers, D., Juárez, M., van Goethem, T., Joosen, W.: Automated website fingerprinting through deep learning. In: Proc. of Network and Distributed System Security Symposium (NDSS) (2018)
37. Saadatpanah, P., Shafahi, A., Goldstein, T.: Adversarial attacks on copyright detection systems. In: Proc. of Int. Conference on Machine Learning (ICML) (2020)
38. Schlegel, R., Zhang, K., Zhou, X., Intwala, M., Kapadia, A., Wang, X.: Soundcomber: A stealthy and context-aware sound trojan for smartphones. In: Proc. of Network and Distributed System Security Symposium (NDSS) (2011)
39. Son, W., Cho, H.T., Yoon, K.: Sub-fingerprint masking for a robust audio fingerprinting system in a real-noise environment for portable consumer devices. In: Digest of Technical Papers International Conference on Consumer Electronics (ICCE) (2010)
40. Sonnleitner, R., Widmer, G.: Robust quad-based audio fingerprinting. IEEE ACM Transactions on Audio, Speech, and Language Processing **24**(3) (2016)
41. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research (JMLR) **15**(1), 1929–1958
42. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
43. The New York Times Company: All 3 billion yahoo accounts were affected by 2013 attack. https://www.nytimes.com/2017/10/03/technology/yahoo-hack-3-billion-users.html (2017), (last visited Apr. 22, 2024)
44. The New York Times Company: That game on your phone may be tracking what you're watching on TV. https://www.nytimes.com/2017/12/28/business/media/alphonso-app-tracking.html (2017), (last visited Apr. 22, 2024)
45. Thiemert, S., Nürnberger, S., Steinebach, M., Zmudzinski, S.: Security of robust audio hashes. In: IEEE International Workshop on Information Forensics and Security (WIFS) (2009)
46. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is All you Need. In: Advances in Neural Information Processing Systems (NIPS) (2017)
47. Wang, A.: An industrial strength audio search algorithm. In: Proc. of the Int. Society for Music Information Retrieval Conference (ISMIR) (2003)
48. Wang, A.: The shazam music recognition service. Commun. ACM **49**(8) (2006)
49. Warden, P.: Speech commands: A dataset for limited-vocabulary speech recognition. CoRR **abs/1804.03209** (2018), http://arxiv.org/abs/1804.03209

50. White, A.M., Matthews, A.R., Snow, K.Z., Monrose, F.: Phonotactic reconstruction of encrypted voip conversations: Hookt on fon-iks. In: Proc. of IEEE Symposium on Security and Privacy (S&P) (2011)
51. Xu, Y., Frahm, J., Monrose, F.: Watching the watchers: Automatically inferring TV content from outdoor light effusions. In: Proc. of ACM Conf. on Computer and Communications Security (CCS) (2014)
52. Zapr Media Labs: Privacy | Zapr Media Labs[8] `https://tinyurl.com/rneknwyb` (2022), (last visited Apr. 22, 2024)
53. Zapr Media Labs: Zapr | TV analytics, integrated advertising, real-time surveys[3]. `https://tinyurl.com/2vhr6vmu` (2022), (last visited Apr. 22, 2024)
54. Zimmeck, S., Li, J.S., Kim, H., Bellovin, S.M., Jebara, T.: A privacy analysis of cross-device tracking. In: Proc. of USENIX Security Symposium (2017)

A Details on ACR Solutions

In this section, we provide information that we obtained through reverse engineering of the two commercial ACR solutions Zapr and ACRCloud.

Analysis Setup

We start by describing our experimental setup to reverse engineer the apps and discuss our findings of both solutions afterward.

Mobile Apps. For ACRCloud, we base our analysis on the Deezer app (version 6.1.14.99.) and verify that our insights also remain valid for more recent versions of the SDK (version 6.2.13.151). Similarly, we use the Android smartphone application *ABP Live TV News* (version 9.9.7) for Zapr.

Dynamic Analysis. Both solutions encapsulate the implementations of the ACR algorithms in a shared library, which is provided as a native binary object and accessed by the Android apps through the Java Native Interface (JNI). We treat the fingerprinting algorithms inside the shared object as a black box and observe its return values. To this end, we use the dynamic instrumentation toolkit *Frida*, which allows us to run the fingerprinting algorithms on controlled input signals, and extract the resulting audio fingerprints. A static analysis shows that all algorithms expect the input signal to be sampled at a frequency of 8,000 Hz with an audio bit depth of 16 bit. Providing the ACR implementations with properly preprocessed audio samples yields the required audio fingerprints, which can then be utilized for further analysis.

To learn more about the underlying structure of the fingerprints, we perform controlled experiments using specifically crafted audio signals from which we derive audio fingerprints. For instance, we use audio signals that contain only one particular frequency or even pure silence.

[8] Zapr discontinued its service in mid 2022. Thus, we can only provide a link to the snapshot of the website.

Table 2: Overview of final model parameters. The table shows the number of bits of the subfingerprints (l), sequence length (k), the embedding size (d), the number of encoder blocks ($encoders$), and the number of heads per encoder ($heads$) for each setting.

Experiment	Fingerprint	l	k	d	$encoders$	$heads$
Speech vs. Music	QBF	19	177	128	4	4
	ACRCloud	64	96	128	4	4
	Zapr Alg1	32	80	128	4	4
	Zapr Alg2	8	804	128	4	4
Speaker Identification	QBF	19	118	128	4	4
	ACRCloud	64	95	128	4	4
	Zapr Alg1	32	80	128	4	4
	Zapr Alg2	16	314	128	4	4
Word Identification	QBF	19	48	128	4	4
	ACRCloud	64	24	128	4	4
	Zapr Alg1	32	20	128	4	4
	Zapr Alg2	32	92	128	4	4

Fingerprint Structures

We find that the fingerprint structures do not only widely differ between ACRCloud and Zapr, but even between the two Zapr algorithms we selected for our analysis. In the following, we provide more details on our findings.

ACRCloud. For ACRCloud, we find that the generated audio fingerprints vary in length, although all audio snippets are three seconds long. In particular, the length of the generated fingerprints for our dataset varies between 344 and 752 bytes, with a median at 544 bytes. Each fingerprint consists of multiple subfingerprints (see Section 3.2) with a length of 8 bytes. The first two bytes of each subfingerprint encode the frequency of an identified peak. Here, ACRCloud seemingly segments the frequency band, which has a maximum frequency of 4,000 Hz, into 1024 distinct bins of equal size, leading to a frequency resolution of $f_{res} = \frac{4000\ Hz}{1024} \approx 3.906\ Hz$. The third and fourth byte of the subfingerprints encode the time offset Δt with a granularity of roughly 20 ms. For the last four bytes, we are unable to derive clear explanations. But we notice that the information stored in these bytes depend on the frequency bytes but not on the time offset.

Zapr. For Zapr Alg1, none of the observed fingerprints exceeds 340 bytes in length, which suggests a maximum length for the fingerprints. Additionally, each fingerprint's length is divisible by 4 bytes, indicating that they are composed of multiple subfingerprints, each 4 bytes long. The only exception we find is for silent signals, for which the algorithm does not output any fingerprints. The first two bytes of a fingerprint encode the time offset with a precision of 2 seconds. The last byte encodes frequency information, systematically partitioning the 4 kHz-band into 256 distinct frequency bins. The purpose of the third byte remains unclear. Unfortunately, for Zapr Alg2, we have not been able to derive information about its structure.