Fidelity Leakages: Applying Membership Inference Attacks to Preference Data

Pierre Danhier, Clément Massart, François-Xavier Standaert
Crypto Group, ICTEAM Insititute, UCLouvain, Belgium

***Abstract*—We apply membership inference attacks in the context of preference data exploited by recommendation systems and show that they can lead to "fidelity leakages". These leakages allow one service provider to determine whether or not its users are faithful. We first provide experimental results based on real-world data made available by Spotify that confirm the feasibility of such attacks and allow us to put forward their influencing parameters. We then discuss the challenges for interpreting and mitigating fidelity leakages.**

1. Introduction

Membership Inference Attacks (MIA) aim at detecting if some individuals' data has been used to estimate a statistical model (e.g., to train a machine learning algorithm). By exploiting the dependencies between the training data and the model results, they may involuntarily leak some information about individuals. MIA have attracted a growing attention over the last years, due to a large panel of potential applications exposing privacy risks. For example, in 2008, Homer et al. described how to determine whether or not an individual's DNA data was part of a modeling phase [6]. They experimentally showed the feasibility of MIA based on the distance between the individuals' data and the DNA-based estimated model. More recently, Backes et al. showed that such attacks also threaten the privacy of individuals contributing their microRNA expressions to scientific studies [1]. Other works investigated MIA in the context of machine learning algorithms and classification data, e.g., [9], [11], [3], [4]. Pyrgelis et al. specifically studied their impact for location privacy [8]. Finally, a few papers focused on protection mechanisms [10], [7].

From a more formal viewpoint, many of these works also point out the difficulty to rigorously evaluate MIA [11], [3], [5], [12], [7]. One reason is that the feasibility of MIA is tightly connected to the presence of outliers in the data and the risk of overfitted models (both issues being generally hard to assess). Another problem is that some types of false positives are also impossible to avoid in MIA. For example, two individuals leading to exactly the same data could lead to conclude that one individual's data was used for training a model while it was his twin's data that was used.

The contributions of this paper are twofold.

Our first contribution is to apply MIA in the specific context of recommendation systems, where they can lead to so-called fidelity leakages. Say that two song providers (e.g., Spotify and a competitor) embed a recommendation system. Spotify's goal is to determine whether some of its users are also registered with its competitor. For this purpose, it can use the data of its users to challenge the competitor's recommendation system and mount a MIA. In case of success, it can conclude that some users may utilize both services and decide to apply a special treatment to them.

We illustrate the theoretical feasibility of such attacks against real-world data by exploiting a database made available by Spotify. For this purpose, we split the database in two and consider half the data to be owned by Spotify and half the data to be owned by an hypothetical competitor. We then put forward both concrete risks of fidelity leakages through simple statistical metrics and the usual difficulty to interpret these risks in a mathematically sound manner. We additionally show that these risks are anyway bounded by the accuracy of the competitor service provider's recommendation system (i.e., one cannot perform a MIA with better success than the probability of correct recommendation).

Our second contribution is to give a more quantitative turn to our analyzes, and to discuss the challenges in ensuring actual users that none of the service providers they are registered with can exploit fidelity leakages. For illustration, let us consider Spotify as the adversarial service provider and its competitor as the target one. In this case, users essentially want to be convinced that the data used by the target service provider to estimate its recommendation system is sufficient (i.e., leads to a well generalized statistical model) so that the dependencies between the training data and the model results are not exploitable. Previous works such as the one of Song et al. already showed that the statistical distance between the training data and test data is a good indicator of the feasibility of MIA [10]. We push this intuition one step further and describe a heuristic solution in order to approximate the number of songs collected per user that can rule out the possibility that Spotify exploits fidelity leakages thanks to simple information theoretic quantities.

One important consequence of these investigations is that without additional privacy enhancing mechanisms (e.g., mixnets), preventing such MIA would require users to trust that Spotify limits the amount of data that can be linked to each of them within small (pseudonymized) "sessions", so that users with a lot of data appear as multiple (unlinkable) sessions to Spotify. Furthermore, the maximum session size depends on the generalization of the target's model. So mitigating fidelity leakages in this way requires an unlikely interaction between the different service providers a user is registered with, considering that each service provider can play the role of both an adversary and a target.

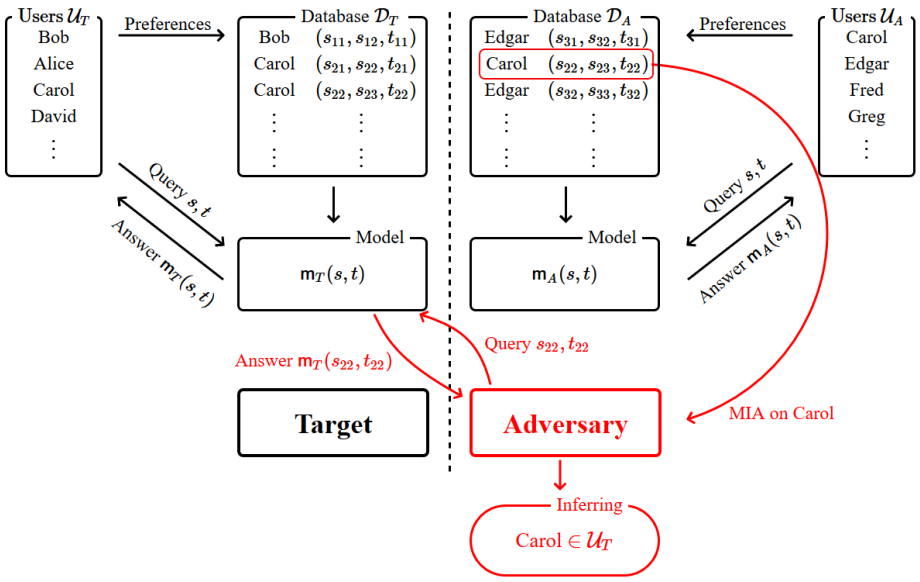

Figure 1. The fidelity leakages' threat model: high-level view.

2. Notations

We first define a set of users as $\mathcal{U} = \{u_1, u_2, \ldots, u_{n_u}\}$, with n_u the number of users. We denote random variables with uppercase letters (e.g., U), sets with calligraphic ones (e.g., $\mathcal{U}$) & events with lowercases (e.g., u). In our investigations, we consider the users of some service provider – we will take the case of a music provider embedding a recommendation system as running example. The service provider has a database of songs $\mathcal{S} = \{s_1, s_2, \ldots, s_{n_s}\}$, with n_s the number of songs. In this context, recommendations to users are based on analyzing the transition between songs for each user. Therefore, we define the j^{th} observations from user u_i as $\boldsymbol{o}_{ij} = \{s_{ij}, s_{ij+1}, t_{ij}\}$. When clear from the context we use the subscript i to denote a link to user u_i. Concretely, s_{ij} and s_{ij+1} represent two consecutive songs within a session (i.e., a list of consecutive songs for a given user), which are usually (but not mandatorily) different; t_{ij} correspond to complementary timing data about the users' behavior while listening to the song s_{ij} (e.g., skipping information, pauses, listening hour). The observations of a user i are then contained in a set $\mathcal{O}_i = \bigcup_{j=1}^{n_i} \boldsymbol{o}_{ij}$, with n_i the number of observations from user i. Finally, the database of all the observations is defined as $\mathcal{D} = \bigcup_{i \in \mathcal{U}} \mathcal{O}_i$.

3. Threat model

Our threat model is depicted in Figure 1, where the left and right parts of the figure represent two service providers. One is the adversary (trying to infer the fidelity of its users), the other one is the target. The adversary owns a subset of users denoted as $\mathcal{U}_A \subseteq \mathcal{U}$. The target service provider also owns a subset of users denoted as $\mathcal{U}_T \subseteq \mathcal{U}$. The adversary aims at finding the users from $\mathcal{U}_A$ that are also present in $\mathcal{U}_T$ (i.e., users in $\mathcal{U}_A \bigcap \mathcal{U}_T$). We call the detection of such users a fidelity leakage, since it allows the adversary to apply a special (positive or negative) treatment to those unfaithful users (e.g., in order to keep them as clients).

To detect the fidelity of users, the adversary has access to a few objects. The first one is the database of observations of its own users $\mathcal{U}_A$ denoted as $\mathcal{D}_A = \bigcup_{i \in \mathcal{U}_A} \mathcal{O}_i$. Note that a similar database, $\mathcal{D}_T$ from $\mathcal{U}_T$ is possessed by the targeted service provider but is not directly accessible by the adversary. The other objects are the statistical models used for recommending songs. Namely, both service providers have a public recommendation system to suggest next songs to their respective users based on a previous song (and possibly some complementary timing data). We next denote them as $\mathsf{m} : (s, t) \mapsto s'$, with $s, s' \in \mathcal{S}$ the previous and next songs and t the timing data. We further use m_A for the adversary's model which is trained on the set $\mathcal{D}_A$, and m_T for the target's model which is trained on the set $\mathcal{D}_T$. To make explicit the fact that a user comes from $\mathcal{D}_A$, we denote the j^{th} observation from user i that belongs to $\mathcal{U}_A$ as $\boldsymbol{o}^A_{ij} = \{s^A_{ij}, s^A_{ij+1}, t^A_{ij}\}$. So the adversary has access to his database $\mathcal{D}_A$, his model m_A and the target model m_T via queries. For all $\boldsymbol{o}^A \in \mathcal{D}_A$, he can query the target's model to obtain responses $\mathsf{m}_T(s^A, t^A)$. Based on these responses and for each of his users, he can try to infer whether or not there is a chance that the user is also part of $\mathcal{U}_T$.

4. Simple evaluation metrics

Fidelity leakages occur in case a successful MIA can be applied to preference data. For any user $u \in \mathcal{U}_A$, the goal of the adversary is to infer whether this user is in the competitor's set $\mathcal{U}_T$, leading to the four possible success/failure cases included in Table 1. Informally, the True Positive (TP) rate is the proportion of users correctly reported as an unfaithful ones; the False Positive (FP) rate

	Reality		Guess	
Success	$u \in \mathcal{U}_A \bigcap \mathcal{U}_T$	$\rightarrow$	$u \in \mathcal{U}_T$	True Positive (TP)
	$u \notin \mathcal{U}_A \bigcap \mathcal{U}_T$	$\rightarrow$	$u \notin \mathcal{U}_T$	True Negative (TN)
Failure	$u \in \mathcal{U}_A \bigcap \mathcal{U}_T$	$\rightarrow$	$u \notin \mathcal{U}_T$	False Negative (FN)
	$u \notin \mathcal{U}_A \bigcap \mathcal{U}_T$	$\rightarrow$	$u \in \mathcal{U}_T$	False Positive (FP)

TABLE 1. POTENTIAL FIDELITY LEAKAGES' SUCCESS & FAILURES.

is the proportion of users wrongly reported as unfaithful ones; the True Negative (TN) rate is the proportion of users correctly reported as faithful ones; finally, the False Negative (FN) rate is the proportion of users wrongly reported as faithful ones. We say that an attack is a potential success in case of true positive or negative. Otherwise it is a failure. So overall, the attack's (potential) success rate is the proportion of users that are correctly classified by the adversary.

As will be detailed next, we use the term potential success because the interpretation of these metrics is difficult. For example, a true positive can generally be due to both a fidelity leakage and to a well generalized target model.

We note that the impact of a failure is quite different depending on the cases. For example, a false positive typically implies a risk to apply a special treatment to a faithful user (so a financial loss in case of a commercial offer). It can happen if two different individuals present in $\mathcal{D}_A$ and $\mathcal{D}_T$ have very similar preferences and therefore observations. By contrast, a false negative rather comes with a risk that an unfaithful user leaves the service in favor of a competitor.

We note also that the concrete estimation of these metrics requires an omnipotent evaluator who can access both the adversary and the target's databases. Next, they will be evaluated in function of the size of the target database, that an evaluator can make vary from the maximum size to arbitrarily small ones by simply dropping observations.

5. Experimental setup

We now describe how we evaluated the threat model of Section 3 in a concrete setting. Since, as just mentioned, the estimation of our metrics requires an omnipotent evaluator, we analyzed the data of a single service provider that we split in two equal parts to emulate two databases $\mathcal{D}_A$ and $\mathcal{D}_T$. The main advantage of this emulation is that it also allows us to control the proportion of shared users between the two databases, which is an important attack parameter. The impact of this emulation is discussed in Section 8.

5.1. Spotify/CrowdAI dataset

The data we investigated comes from the music service provider Spotify. Its users can access a large panel of songs and listen to playlists they make or that are made by other users. It offers various options such as shuffling songs, skip songs, ... The users' preferences and listening behavior are recorded in a large database. In order to improve its recommendation system, Spotify published a pseudonymized version of a small part of its database on the CrowdAI platform (https://www.crowdai.org/challenges/spotify-sequential-skip-prediction-challenge). To protect the privacy of the users, the dataset contains a list of sessions with up to 20 songs each, so that it is not possible to link multiple sessions to a single user. In our experiments, we consider each session as originating from a single user. From each session, we get a list of songs (so the transitions between them) and complementary timing information.

The original data set contains around 125 millions sessions and 2 billions listenings. For simplicity, we only kept full sessions (i.e., with 20 songs) including at least 16 songs listened entirely. We also focused on sessions that came from personal playlists which we expected to be more reflective of user preferences. We finally reduced the number of different songs considered (so that we have a sufficient number of transitions observed for all the songs). This number was set to 229 based on the ratio between the number of sessions and the number of songs, leading to an exploitable database of 11,156 sessions for a total of 223,120 listenings. We note that this data preprocessing was only aimed to obtain a good rate of correct predictions for our recommendation system, as would be the goal of an actual system provider.

5.2. Recommendation system

In order to make the threat model of Section 3 concrete, we need to choose a statistical model for the target recommendation system: its goal is to predict a transition (i.e., a next song based on a previous song and the timing information). In practice, we used a Random Forest model implemented in the scikit-learn Python library (https://scikit-learn.org/stable/), and we set the number of decision trees parameter to 20. This choice is motivated by its simplicity (of both implementation and interpretation).[1]

5.3. Metrics estimation

We estimate our metrics by directly challenging the target model m_T with observations from the adversary's database $\mathcal{D}_A$. Whenever there is a match between the target model's prediction and the adversary's data for a user u_i, the adversary considers the transition as coming from a potentially unfaithful user. This corresponds to the "Guess" part of Table 1. The guess is then classified as a true positive, true negative, false positive or false negative by checking whether the user is in the target database $\mathcal{D}_T$ (which is only possible if the evaluator has full access to this database).

6. Experimental results

We next apply our threat model and metrics to the data described in the previous section. We start with a preliminary evaluation of our prototype recommendation system and then discuss fidelity leakages. In all our MIA experiments, we considered two parameters. First, the size of the target database (measured in number of 20-song sessions), which is the main parameter influencing the generalization

1. Preliminary experiments launched with other statistical tools and parameters did not significantly affect our main conclusions.

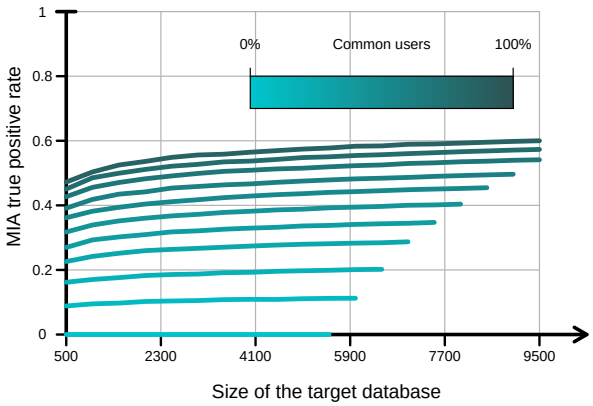

Figure 2. Potential fidelity leakages: true positives' rate.

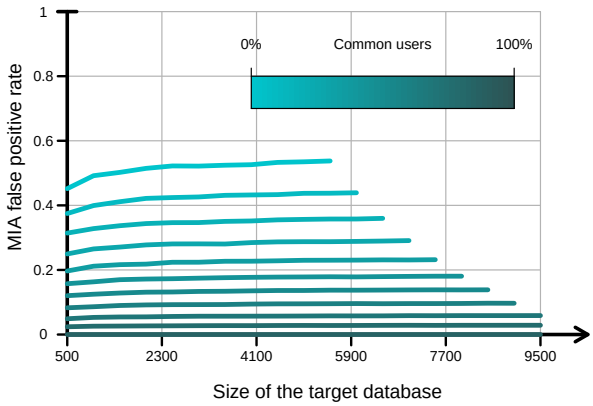

Figure 3. Potential fidelity leakages: false positives' rate.

of the target model. Second, the amount of common users in $\mathcal{D}_A$ and $\mathcal{D}_T$, which would be fixed & unknown by the adversary in a real attack, and that we can vary in our emulated databases (to improve the understanding of the results).

6.1. Preliminaries

Before analyzing fidelity leakages, we evaluate the quality of our recommendation system based on random forests. We estimated the rate of correct predictions that our recommendation system reaches by leveraging 10-fold cross-validation (9/10th of the data used to build the model, 1/10th of the data used to test it). Concretely, we reach a prediction rate of approximately $\approx 60\%$ (which is to be compared with a random guess of $\frac{1}{229}$). We further observed that the fraction of common users does not significantly impact the accuracy of the recommendation system, as expected. Besides, we checked that further increasing the database size would not significantly improve this prediction rate (which already saturates when all observations are used).

6.2. Fidelity leakages

We now move to the estimation of the simple metrics introduced in Section 4. We start with the analysis of true positives, which is the most concretely-relevant metric to evaluate, since it directly impacts the commercial answer a service provider could impose to its users. The results of this experiment are given in Figure 2, which leads to two important observations. First, the true positive rate increases in function of the fraction of common users: the more common users, the more fidelity leakages can be detected. Second, this true positive rate saturates to $\approx 60\%$, which actually corresponds to the accuracy of the target's recommendation system. The latter is interesting and actually intuitive: since fidelity leakages essentially exploit the similarities between the target's model and adversary's observations, it requires that the target's model is sufficiently predictive of the users' true preferences in the first place. For example, a recommendation system outputting random songs would not allow detecting fidelity leakage better than a random guess.

The false positive rate in Figure 3 provides a complementary view to Figure 2 . It is also impacted by the percentage of common users, but in an inverse manner: more common users reduce the amount of false positives. A concrete adversary would typically like to minimize this metric since it may lead to wasted promotional offers.

Note that the steps in these two Figures' curves are only due to the (technical) way we emulate $\mathcal{D}_A$ and $\mathcal{D}_T$ (i.e., we have more observations when more users are shared).

Interpretation. The experiments in this section confirm the theoretical feasibility of MIA against recommendation systems that can lead to fidelity leakages. Yet, as usual with MIA, the interpretation of the previous (simple) evaluation metrics is difficult. In particular, it is in general hard to determine whether (for example) a true positive in Figure 2 is due to an unfaithful user or to a well generalized model. Typically, a concrete adversary would have a stronger suspicion of unfaithful user for such a true positive if it occurs for an outlier observation of his database $\mathcal{D}_A$ which, as mentioned in introduction, is in general hard to assess rigorously. This is why we refer to these results as potential fidelity leakages (which is the best that can be easily estimated). Motivated by this hard to assess risk, we next investigate how simple information theoretic tools can give a more quantitative turn to our analyses, and serve as a better basis in order to discuss the main challenges in preventing fidelity leakages.

7. Mitigating fidelity leakages

Following the previous caveats, an important intuition behind all MIA is that their interpretation depends on the convergence of the statistical model (e.g., recommendation system) of the target service provider, which depends on the size of its database. We next investigate the possibility to use this intuition in order to bound the risk that Spotify can detect fidelity leakages by querying its competitor's recommendation system. In this respect, a starting point is the fact that MIA exploit the correlation between the adversary's observations in $\mathcal{D}_A$ and a target model m_T estimated from $\mathcal{D}_T$. So in order to bound the risk that such

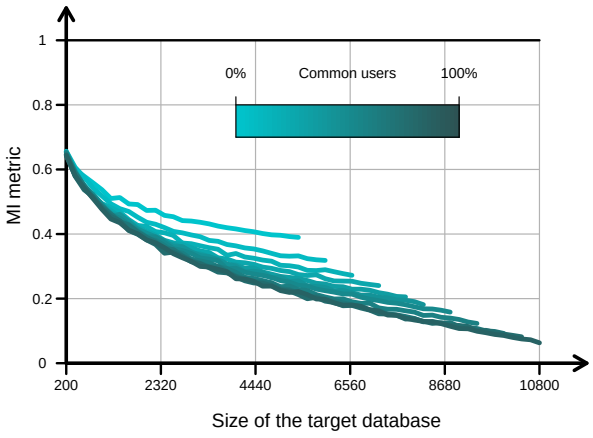

Figure 4. Evolution of $\hat{\mathrm{MI}}(H;O)$.

attacks succeed, a natural approach is to bound the statistical distance between the distribution of the observations in $\mathcal{D}_A$ and $\mathcal{D}_T$. Intuitively, if these distributions get close, it implies that the models should better generalize and MIA should become more difficult. This intuition is exactly the one exploited by Song et al. in [10], who used the Kullback-Leibler (KL) divergence as a distance metric and show that a small KL divergence implies harder to mount MIA.

We next use the Mutual Information with a similar motivation, and because it allows us to make results and claims slightly more interpretable and quantitative. For this purpose, we start by defining a new (binary) random variable H which stands for data Holder and can take values A or T. We then define the estimated Mutual Information (MI) between the data holders and their observations as:

$$\hat{\mathrm{MI}}(H;O) = 1 + \sum_{h\in\{A,T\}} \Pr[h] \cdot \sum_{\boldsymbol{o}\in\mathcal{D}_h} \hat{\Pr}[\boldsymbol{o}|h] \cdot \log_2\left(\hat{\Pr}[h|\boldsymbol{o}]\right).$$

Concretely, $\hat{\Pr}[\boldsymbol{o}|h]$ is estimated by building histograms for the observations; $\hat{\Pr}[h|\boldsymbol{o}]$ is computed via Bayes.

A small mutual information implies that given some observations, it is hard to tell whether they come from the distribution of the target or the one of the adversary (so MIA should be hard to mount): $\mathrm{MI}(H;O) = 0$ would formally rule out that such attacks are possible at all. We use this metric because it directly gives an indication of the number of observations that would be needed to tell apart the distributions A and T thanks to Fano's inequality [2]. Namely, the average number of samples needed to reach this goal with good probability is bounded as $N \geq \frac{1}{\mathrm{MI}(H;O)}$.

We report the evolution of the estimated MI in function of the size of the target database in Figure 4. As expected, it is decreasing with the size of the database. Indeed, as this size increases the distributions of the observations estimated from $\mathcal{D}_A$ and $\mathcal{D}_T$ become more and more similar.[2] In other words, asymptotically the data of both service providers should tend towards the same distribution which reflects all user's true preferences. As a side-remark, it is worth observing that the fraction of common users influences the estimated MI values. Yet, this time it only does it because increasing the fraction of common users increases the size of the target database. Asymptotically, the MI should tend towards zero independent of this fraction. So the main security parameter is the size of the target database.

Interpretation. We now discuss the heuristic interpretation of the previous results (a discussion of its theoretical limitations comes in the next section). For this purpose, let us assume that the size of the target database is such that we observe an estimated MI of 0.2, meaning that a set of 10 (random) observations should be enough to distinguish the distributions of $\mathcal{D}_A$ and $\mathcal{D}_T$. Let us further assume that the adversary has more than 10 observations for a user for which he wants to challenge the fidelity. Two cases can happen:

1) The adversary can attribute these observations to his distribution only (i.e., he can distinguish the two distributions with confidence). Then, he can conclude that this user is not unfaithful with the target competitor.
2) The adversary cannot attribute these observations to a single distribution. Then, (*i*) he cannot rule out the possibility of an unfaithful user, but (*ii*) he cannot decide whether his failure is due to a lack of fidelity or a well generalized model with confidence either.

Eventually, the limitations of this approach remain close to the ones of the simpler metrics in the previous section: it is hard to interpret a true positive, which may be due to fidelity leakages or to model generalization. The main (positive) difference is that it is now possible to conservatively guarantee that no fidelity leakages are possible up to a certain number of observations collected per user. For this purpose, it is (roughly) sufficient to require that the estimated MI is small enough, so that the bound on N is large enough.

8. Discussion & Conclusion

We finally discuss our results in more general terms, together with assumptions and interpretation issues.

Outliers' behaviors. Our heuristic analysis puts forward risks of fidelity leakages. Yet, the MI is an average metric and it is expected that user dependencies will be observed in practice. Interestingly, and as suggested in Section 6.2, the presence of outliers is generally going to be an advantage for the adversary service provider, increasing the practical relevance of the MIA threat model. Typically, challenging the target recommendation system with the observations of users with "non-average" behavior increases the chance that a true positive is due to an unfaithful user rather than a generalized model. Unfortunately, this cannot be analyzed with the data released by Spotify, since the observations are only available in short (20-song) pseudonymized sessions (i.e., not enough to detect an outlier behavior with a sufficient number of observations, so that our probabilistic approach

2. The MI with 100% of common users is not equal to zero because we stopped its evaluation before using all the data (i.e., before the trivial situation where $\mathcal{D}_A = \mathcal{D}_T$). For smaller amounts of data, the sets $\mathcal{D}_A$ and $\mathcal{D}_T$ are randomized which allows better reflecting model generalization.

applies). However, the general trend to store more and richer data reflecting users' behaviors in various contexts can only increase the privacy concerns that MIA raise.

Impact of our emulated setup. The assumptions that both the adversary's and the target's databases and models are converging towards the same ground truth is admittedly simplifying. In practice, it is expected that their sampling may remain (at least slightly) different, possibly making the attacks more challenging to perform. Yet, in view of the large amounts of data collected by current recommendation systems (see next), it seems difficult to rule out the risk of fidelity leakages based on this argument. Similarly, the risk of fidelity leakages is increasing with the quality of the target service provider's recommendation system, but this is in line with its natural incentive (to please its users).

Countermeasures. Based on the exhibited risks, it seems natural that users could want to be convinced that no fidelity leakages can be exploited against them. Taking the example of Spotify, a first step in this direction is to be convinced that it cannot leak fidelity information to competitors – which is a natural goal for any service provider. For this purpose, a solution is to perform exactly the self-evaluation of Section 7 and to approximate the number of observations per user N that would prevent any fidelity leakage to be exploited (possibly taking outliers into account). The challenge is that Spotify should then convince its users that no competitor has more than N observations about them. For this purpose, a direct solution would be that the competitors store their data as small (pseudonymized) sessions, so that users with a lot of data appear as multiple (unlinkable) sessions to the competitors themselves. Furthermore, the opposite situation should also be satisfied: the competitors should perform a self-evaluation of their model generalization, which would then impose a maximum session size to Spotify. As mentioned in introduction, such an interaction seems unlikely. Limiting the storage to small sessions (yet sufficient to build an efficient recommendation system) however appears as a good practice, even if not perfectly quantified thanks to constructive interaction. Alternatively, avoiding trusting system providers to behave in such a privacy-respecting manner would require the implementation of mixnets, allowing users to determine the length of the sessions they want to allow.

We note that admittedly, the (significant) overheads that both types of solutions imply are in good part due to the theoretical difficulty to separate fidelity leakages from good generalization, which for now seems inherent to MIA.

Data asymmetries. Since the efficiency of MIA mostly depends on the amount of data accumulated by service providers, it is also worth underlining that data asymmetries (i.e., adversaries with significantly more data than their potential targets) make the risk of fidelity leakages significantly more critical. On the one hand, an adversary with a lot of data should have facilities to identify "outlier users" and also has a lot of data per user (so potentially very large N values) to attack. On the other hand, a target with less data will have stronger incentives to aggressively exploit its data to optimize its recommendation system, which should increase the risk of model overfitting and therefore fidelity leakages. Taking our case study of music recommendation, published figures suggest that established players (e.g., such as Spotify) enjoy such a dominating position.[3]

Conclusion. Conceptually, our results put forward the possibility that service providers exploit fidelity leakages against their users, thanks to a new application of MIA. We believe this is an important privacy concern since fidelity is not something that users want to leak. Technically, our results put forward a significant gap between the concrete peculiarities that make the attack results difficult to interpret and the difficulty to obtain strong security guarantees against such attacks. On the one hand, MIA results can only reveal *risks* of fidelity leakages and it is not possible (at least with existing tools) to separate such leakages from a good model generalization with high confidence. On the other hand, making sure that such risks cannot be exploited requires strong limitations on the service providers' side (i.e., a limit on the number of observations that can be linked to each user) or more control on the user's side (i.e., mixnets).

Acknowledgments. ERC Consolidator Grant 724725. Belgian Fund for Scientific Research (FNRS-F.R.S.).

References

[1] M. Backes, P. Berrang, M. Humbert, and P. Manoharan. Membership privacy in microrna-based studies. In *ACM CCS*, pages 319–330. ACM, 2016.

[2] T. M. Cover and J. A. Thomas. *Elements of information theory (Second Edition)*. Wiley, 2006.

[3] J. Hayes, L. Melis, G. Danezis, and E. D. Cristofaro. LOGAN: membership inference attacks against generative models. *PoPETs*, 2019(1):133–152, 2019.

[4] B. Hilprecht, M. Härterich, and D. Bernau. Monte carlo and reconstruction membership inference attacks against generative models. *PoPETs*, 2019(4):232–249, 2019.

[5] N. Li, W. H. Qardaji, D. Su, Y. Wu, and W. Yang. Membership privacy: a unifying framework for privacy definitions. In *ACM CCS*, pages 889–900. ACM, 2013.

[6] N. Homer et al. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLOS Genetics*, 4(8):1–9, 08 2008.

[7] M. Nasr, R. Shokri, and A. Houmansadr. Machine learning with membership privacy using adversarial regularization. In *ACM CCS*, pages 634–646. ACM, 2018.

[8] A. Pyrgelis, C. Troncoso, and E. D. Cristofaro. Knock knock, who's there? membership inference on aggregate location data. In *NDSS*. The Internet Society, 2018.

[9] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *IEEE S&P*, pages 3–18. IEEE Computer Society, 2017.

[10] L. Song, R. Shokri, and P. Mittal. Membership inference attacks against adversarially robust deep learning models. In *IEEE S&P*, pages 50–56. IEEE, 2019.

[11] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei. Towards demystifying membership inference attacks. *CoRR*, abs/1807.09173, 2018.

[12] Y. Long et al. Understanding membership inferences on well-generalized learning models. *CoRR*, abs/1802.04889, 2018.

3. https://www.businessofapps.com/data/spotify-statistics/