University of London
Imperial College of Science, Technology and Medicine
Department of Computing

Evaluating privacy and robustness in modern data processing systems

Ana-Maria Creţu

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of the University of London and
the Diploma of Imperial College, February 2023

Abstract

Modern technologies collect, process, and share our data on an unprecedented scale. Understanding how these technologies affect our privacy before they are deployed is a key question. This involves taking an adversarial approach to evaluate privacy. Privacy attacks evaluate the robustness of technologies to adversaries aiming to learn sensitive information about individuals. In this thesis, we target technologies whose risks are not well understood in practice, such as anonymisation techniques for dynamic datasets, query-based systems (QBS), machine learning (ML) models, and on-device client-side scanning (CSS) for illegal content detection. We argue that there is a need to develop new threat models and methodologies to comprehensively evaluate their risks in practice. We make five contributions, proposing new threat models and attacks.

Our first contribution is a study of the robustness of anonymisation techniques relying on pseudonymisation and frequent re-pseudonymisation for interaction data, showing that people's interaction data can be accurately identified in large-scale populations even after multiple weeks. We then turn to aggregation systems, which are believed to be safer than releasing individual-level data. Our second contribution is to develop a method to automatically discover privacy vulnerabilities in QBSs that matches or outperforms previous expert-designed attacks. Third, we study a new type of leakage in ML models, the leakage of correlations between the input attributes of a tabular training dataset, showing that correlations can be used as building blocks for stronger attribute inference attacks. Fourth, we study white-box membership inference attacks, analysing the impact of misalignment, a known characteristic of deep neural networks stemming from their weight symmetries, on their performance. Finally, we develop detection avoidance attacks against very recently proposed perceptual hashing-based CSS systems, in order to evaluate whether their privacy risks can be balanced by effective detection.

Acknowledgements

I would like to thank my supervisor, Yves-Alexandre de Montjoye, for encouraging me to pursue in depth a broad range of ideas, for steering me to refine my research towards more fundamental questions, and for teaching me how to communicate science both in writing and orally.

I would like to thank my examiners, Vitaly Shmatikov and Emil Lupu, for their feedback and suggestions on this thesis, which have helped me improve it and gain perspective.

I would like to thank all my Imperial collaborators: Federico Monti, Xiaowen Dong, Michael Bronstein, Stefano Marrone, Antoine Cully, and Andrea Gadotti. I am immensely grateful to Florent Guépin, Florimond Houssiau, and Shubham Jain for all the time spent together working on our papers. I would like to thank Luc Rocher for providing various rounds of feedback on my first paper: I still remember when you told me – emphatically – that it is good enough to send, which greatly encouraged me. I am also grateful for the opportunity to collaborate with and mentor Vince Guan, Matthieu Meeus, and Bozhidar Stevanoski, and to have co-supervised the master theses of the following students: Guilherme Freire, Manuel Fay, Florent Guépin, Miruna Rusu, Cyril Miras, and Hongji Wu.

I would like to thank Thorsten Strufe for hosting me as a visitor at Karlsruhe Institute of Technology and his research group for their warm welcome.

The journey leading to this PhD really started when I left Romania as an 18-year old to study mathematics and discover the world. I dreamed of doing something great, and at that time there was nothing greater to my eyes than doing mathematics. I owe the opportunity to study abroad first, to my highschool French teacher Ioana Voinea, who told me about the competitive scholarship that allowed me to study in France, and second, to the Odon Vallet foundation that paid for the scholarship.

This journey would not have possible without the dear ones who have believed in me during all these years and the support of several people who have guided my path.

First, I am immensely grateful to my math teacher from Romania, Gabriela Toader, who kindled my interest in mathematics in the 5th grade (and to my brother who was instrumental in this, you know how). I will always fondly remember the times spent solving geometry problems and how much I enjoyed it. Working with you has shaped my attitude to solving questions and,

today, doing research: with a passion for mathematics, rigour, and doubt. You have been a great role model to me.

Second, I am grateful to my math teachers from the French Classes Préparatoires: Anne-Laure Biolley and Yves Duval, who taught me a *lot* of mathematics from scratch (that is, starting from only a few axioms), as is the tradition in the French math curriculum. This has fed by curiosity and passion for learning in a way that I did not even imagine was possible.

Third, I am grateful to the numerous mentors I have had along the way, who have taught me numerous skills and shaped me into a better computer scientist. In order: to Rodrigue Tandu, who taught me about leadership, empathy, and authority. To Anne-Laure Derepas, who welcomed me to Klee Group and supported me in my internship project. To Sertan Girgin, who taught me how to code with patience, attention to details, and how to thoroughly optimise my code. The efficiency and extent of the empirical evaluations of this thesis can be traced back to the skills acquired during this internship and the many lessons I have learnt from you. I also fondly remember how generously you welcomed me at Google and how you showed me the whole building and all the nice places I should know of; I always do the same when welcoming new students. To François Fleuret who drew my attention to the importance of deep learning and taught me the art of Pytorch tensors while I was an MSc student at EPFL. To Craig Wright, who introduced me to natural language processing and Word2vec and who gave me autonomy to explore during my internship at Google. To Thomas Lukasiewicz, who welcomed me at Oxford and introduced me to the Winograd Schema Challenge, and to Oana-Maria Camburu, who taught me the fundamentals of deep learning. To Davide Eynard who was immensely kind and provided great support during my internship at Twitter. Finally, to Shruti Tople and Daniel Jones, who generously welcomed me at Microsoft, for your constant optimism and positive, constructive feedback.

Finally, I am reaching the point where words cannot express my love and gratitude towards people who have been on my side, close or far, for a very long time. Still, I will try.

I owe a debt of gratitude to my family, and in particular to my parents and grandparents, for loving me and believing in me all these years. During the hardest of times, being able to conjure up the love, attention, and recognition you have given me, as well as your humour, generosity, integrity, and authenticity has renewed my hope and allowed me to keep going. Thank you, mami, for always reminding me how much you love me, and for all the time you

have given me: listening to me, giving me the space to work on math problems without having to worry about anything else. Thank you for *everything*. Thank you, tati, for all the music you introduced me to, for always believing in me and reminding me how proud you are of me and for helping me (from afar) navigate the world when I was still very young. Thank you, Corina, Ion, Anca, Roxana, Oana, Silviu, Diana, Horia, Petru, Jean-Jil for your friendship and support. Also thank you Artur, Walid, Olivier, Laïla, Claire, Laure, Mathilde, Yixin, Jingtao, Matthieu, Dudu, Weiwei, Radu, Nohemi, Chia-Man, and Dan for all the time we spent together at various moments, and Monica Varagnat and Rodica Chiorean, for supporting me while I was in France. Thank you, Florent, for being a true friend.

Thank you, Florimond, for your unconditional support. We share many loves: for science, mathematics, hearty food, playful and absurd humour, a certain idealism, and books. I have learnt so much from you: how to relax during a PhD, how to have faith that all will turn out just fine, how to play MTG and many other games. Thank you for always being there during the good, but especially the hard times.

Dedication

This thesis is dedicated to my parents, Mihaela-Dana and Victor, and to my brother, Vlad-Florin.

Dedic această teză părinţilor mei, Mihaela-Dana şi Victor, si fratelui meu, Vlad-Florin.

We shall not cease from exploration
And the end of all our exploring
Will be to arrive where we started
And know the place for the first time.

T. S. Eliot, "Little Gidding"

Statement of Originality

I declare that this thesis is composed by myself and that all the work presented in this thesis is my own, except where otherwise stated. I certify that all material that is not my own work, or part of a collaboration, has been properly acknowledged.

Ana-Maria Creţu

Copyright Declaration

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC). Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose. When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes. Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Contents

Abstract **i**

Acknowledgements **i**

1 Introduction **1**

1.1 Motivation 1

1.2 Contributions 5

1.3 Thesis outline 11

1.4 Publications 12

2 Preliminaries **13**

2.1 Individual-level data 13

2.1.1 Definition 13

2.1.2 Pseudonymisation 14

2.1.3 Re-pseudonymisation 16

2.2 Aggregate data 17

2.2.1 Definition 17

2.2.2 Differential privacy 19

3 On the identifiability of interaction datasets across time 22

3.1 Introduction . . . 22

3.2 Attack model . . . 26

3.3 Attack methodology . . . 28

3.3.1 Overview and justification of the method . . . 28

3.3.2 Preprocessing of a k-IIG . . . 39

3.3.3 Embedding of a k-IIG . . . 45

3.3.4 Triplet sampling procedure . . . 47

3.4 Experimental setup . . . 49

3.4.1 Mobile phone interaction dataset . . . 49

3.4.2 Training setup . . . 51

3.5 Results . . . 52

3.5.1 Probability of identification after one week . . . 52

3.5.2 Impact of time delay of auxiliary data and the dataset . . . 53

3.5.3 Impact of time period length of auxiliary data . . . 54

3.5.4 Impact of the population size . . . 55

3.5.5 Post-hoc analysis of features impacting identification . . . 56

3.5.6 Comparison with prior work and baselines . . . 61

3.6 Bluetooth close-proximity data . . . 69

3.7 Discussion . . . 73

3.8 Conclusion . . . 78

4 Automated attacks against query-based systems 79

4.1 Introduction 80

4.2 Background 82

4.2.1 Query-based systems 82

4.2.2 Evolutionary algorithms 84

4.3 Attack model 85

4.3.1 Adversary access to the query-based system 85

4.3.2 Attribute inference attack 86

4.3.3 Auxiliary knowledge on the target dataset 87

4.4 Attack methodology 88

4.4.1 Overview of QuerySnout 88

4.4.2 Search space of solutions 91

4.4.3 Automated learning of a rule G 93

4.4.4 Search space exploration 94

4.5 Experimental setup 100

4.5.1 Datasets 100

4.5.2 Instantiating the auxiliary knowledge 101

4.5.3 Query-based systems 101

4.5.4 Manual attribute inference attacks 104

4.6 Empirical results 106

4.6.1 Attack parameters 106

4.6.2 Real-world systems 107

4.6.3 Non-deterministic systems: SimpleQBS . . . 109
4.6.4 Comparison with random search . . . 112
4.6.5 Impact of the number of queries . . . 113
4.6.6 Impact of search parameters . . . 114
4.6.7 Impact of dataset size . . . 116
4.6.8 Analysis of solutions . . . 116
4.7 Budget-based systems . . . 120
4.7.1 Extending QuerySnout to BBSs . . . 120
4.7.2 DPLaplace . . . 121
4.7.3 Empirical results . . . 124
4.8 Discussion . . . 126
4.9 Related work . . . 129
4.10 Conclusion . . . 130

5 Background on machine learning privacy 132

5.1 Introduction . . . 132
5.2 Inference attack model . . . 134
5.2.1 Setup . . . 135
5.2.2 Adversary goal . . . 135
5.2.3 Adversary access to the model . . . 136
5.2.4 Adversary knowledge . . . 136
5.3 Literature review on inference attacks . . . 138
5.3.1 Privacy attacks . . . 138

5.3.2 Property inference attacks 143

5.4 Open questions 144

6 Correlation inference attacks against machine learning models 147

6.1 Introduction 147

6.2 Correlation inference attack 150

6.2.1 Attack model 150

6.2.2 Adversary knowledge on data distribution 151

6.2.3 Adversary access to the model 152

6.2.4 Correlation inference task 153

6.3 Attack methodology 153

6.3.1 Overview 153

6.3.2 Spherical parametrisation of correlation matrices 154

6.3.3 Model-less attack 155

6.3.4 Sampling correlation matrices under constraints 158

6.3.5 Sampling data given a correlation matrix 162

6.3.6 Model-based attack 164

6.4 Experimental setup 166

6.4.1 Target models 166

6.4.2 Datasets 166

6.4.3 Attack parameters 167

6.5 Results 168

6.5.1 Impact of constraints for $n = 3$ variables 168

6.5.2 Increasing the number of variables n . . . 172
6.5.3 Impact of mitigations . . . 175
6.5.4 Real-world dataset evaluation . . . 177
6.6 Attribute inference attack . . . 179
6.7 Related work . . . 183
6.8 Discussion . . . 184
6.9 Conclusion . . . 188

7 White-box membership inference attacks using re-aligned shadow models 189

7.1 Introduction . . . 189
7.2 Attack model . . . 191
7.3 Background . . . 192
7.3.1 Deep neural networks . . . 192
7.3.2 Symmetries in deep neural networks . . . 194
7.3.3 Impact of misalignment on white-box shadow modelling . . . 195
7.4 Related work . . . 196
7.5 Problem statement . . . 198
7.6 Causes of shadow model misalignment . . . 200
7.6.1 Selected factors . . . 201
7.6.2 Metrics to measure the misalignment . . . 202
7.6.3 Experimental setup . . . 204
7.6.4 Results . . . 205
7.7 Re-aligning the layers . . . 213

7.7.1 Our approach . . . 213
7.7.2 Re-alignment results . . . 215
7.8 White-box membership inference . . . 217
7.8.1 Evaluation setup . . . 218
7.8.2 Datasets and models . . . 220
7.8.3 Results . . . 221
7.9 Discussion . . . 228
7.10 Conclusion . . . 229

8 Detection avoidance attacks against perceptual hashing client-side scanning 231

8.1 Introduction . . . 232
8.2 Perceptual hashing-based client-side scanning . . . 234
8.3 Attack model . . . 235
8.3.1 Detection avoidance attack . . . 235
8.3.2 Adversary access to the perceptual hashing algorithm . . . 237
8.4 Attack methodology . . . 238
8.4.1 Notation . . . 238
8.4.2 Black-box attack . . . 239
8.4.3 White-box attacks for DCT-based hashes . . . 245
8.5 Experimental setup . . . 249
8.5.1 Perceptual hashing algorithms . . . 249
8.5.2 Attack parameters . . . 250
8.5.3 Dataset . . . 251

8.6 Results 254

8.7 Discussion 261

8.8 Related work 268

8.9 Conclusion 269

9 Conclusion **275**

9.1 Summary of thesis achievements 275

9.2 Perspectives 279

Bibliography **281**

List of Tables

3.1 **The list of the 23 features used for the mobile phone interaction data**. We also list the corresponding methods to compute the features using the bandicoot toolbox. A dash symbol (-) indicates that the feature was not computed using bandicoot. 43

3.2 **The list of 19 features used in the post-hoc analysis.** We also list the corresponding methods to compute the features using the bandicoot toolbox. A dash symbol (-) indicates that the feature was not computed using bandicoot. . 59

3.3 **Logistic Regression summary**. For each feature, we report the p-values when all features are used to fit the model **(I)**, respectively when each feature is used as sole predictor variable **(II)**. 61

3.4 **The list of the 16 features used for the Bluetooth close-proximity interaction data.** We also list the corresponding methods to compute the features using the bandicoot toolbox. A dash symbol (-) indicates that the feature was not computed using bandicoot. 71

3.5 **Probability of identification within rank 1 for different scenarios and values of k.** 74

3.6 **Examples of attention weights for various intervals of the normalised entropy.** We randomly select one attention weight vector in each interval ($s = 1$). In all cases, one or two neighbours have an attention weight at least twice as large as the lowest weight(s). 78

4.1 **Comparison between automated and manual attacks against Diffix** on different scenarios and datasets. We report the attack accuracy over 100 target records, averaged over 5 repetitions with standard deviation. 108

4.2 **Comparison between automated and manual attacks against TableBuilder** on different scenarios and datasets. We report the attack accuracy computed over 100 target records, averaged over 5 repetitions with standard deviation. 108

4.3 **Comparison between QuerySnout and the random search.** We report the absolute difference (in percentage points) between the accuracy of QuerySnout and the accuracy of the random search (with mean and standard deviation over 5 repetitions). 113

4.4 **Ablation experiment in the AUXILIARY scenario.** We report the accuracy of the evolutionary search (with mean and standard deviation over 5 repetitions) where we set one of the mutation probabilities to zero, keeping all the others equal to the default value. 116

4.5 **Accuracy against DPLaplace for different values of ε.** We report the mean and standard deviation over 5 repetitions. 125

4.6 **Comparison between QuerySnout and the random search on DPLaplace for different values of ε.** We report the absolute difference (in percentage points) between the accuracy of QuerySnout and the accuracy of the random search (with mean and standard deviation over 5 repetitions). 125

6.1 **Results of our correlation inference attacks on three real-world datasets.** We abbreviate "Communities and Crime" to "C & C" for spacing reasons. 178

6.2 **Comparison between CI-AIA and other attribute inference attacks on Fifa19.** We report the attack accuracy averaged over 1000 runs (with 95% confidence interval). 183

7.1 **Misalignment scores using different metrics** (mean and standard deviation). Model initialisation is the most influential individual factor responsible for misalignment. The classical adversary corresponds to All $\neq$, while the adversary having knowledge of the target model initialisation corresponds to $\neq$ BO, $\neq$ DS, and DD (both are highlighted in **bold**). 208

7.2 **Weight misalignment scores** for different training dataset sizes. In all of the settings, changing the model initialisation leads to misalignment. 211

7.3 **Weight misalignment scores** for different values of the learning rate η and the early stopping patience E. The training dataset size is 50000. Regardless of the setting, changing the model initialisation leads to misalignment of internal layers that is comparable to applying a random permutation to the layer. . . . 212

7.4 **Misalignment scores for different re-alignment techniques** applied to shadow models trained by the classical adversary. 216

7.5 **CIFAR10: Results of white-box MIA using features from individual layers.** We report the mean AUC with 95% confidence interval. We perform attacks using output activations (OA) or gradients (G) of different layers. Our re-alignment techniques (S5)-(S9) successfully recover the membership signal in each of the individual layers, outperforming the naïve classical adversary (S3). . 222

7.6 **Our input activation (IA) features improve the performance of white-box MIAs.** We report results in the (S1) Target dataset (auditor) scenario. . . 222

7.7 **CIFAR10: Results of white-box MIAs using different threat models and re-alignment methods.** 223

7.8 **Texas100: Results of white-box MIAs using different threat models and re-alignment methods.** We show results of attacks using features from the last layer. For brevity, we omit the results of the black-box attack for (S3)-(S8) as they are identical to (S3), due to re-alignment techniques not impacting the output layer. 224

7.9 **Purchase100: Results of MIAs against an MLP using different threat models and re-alignment methods.** 224

7.10 **Results of MIAs using a set-based classifier.** 227

7.11 **CIFAR10: Comparison with Leino et al. [193].** 227

8.1 **Thresholds selected to simulate the client-side scanning system.** 255

8.2 **Estimate of the number of images that would be wrongly detected daily on WhatsApp for different perceptual hashing algorithms and thresholds.** We here consider a database size of $N = 100$K, 4.5B images being shared daily [36], and a prevalence rate of illegal content of 10^{-4}. 258

List of Figures

3.1 **Setup of the behavioural profiling attack. a** An example of a 2-IIG highlighted in the larger graph it comes from. The vertices of the 2-IIG (inside the dashed green circle) are respectively the originating individual (in yellow), 1-hop neighbours (in gray), and the 2-hop neighbours (in purple). In solid lines are the edges that are part of the 2-IIG: all the edges between the originating and 1-hop neighbours; between the 1-hop neighbours; and between 1-hop and 2-hop neighbours, but excluding those between 2-hop neighbours (dotted lines). Dashed lines are all the other edges. For simplicity, all edges are shown as a single directed edge of thickness proportional to the total number of interactions. **b** The data available to the adversary consist of (left) 2-IIGs coming from time period $[t_1, t'_1)$, usually as part of an anonymised dataset, and (right) auxiliary 2-IIG data about a target individual A ($G^2_{i_0,[t_2,t'_2)}$). While we here display auxiliary data coming from a later period in time, our attack applies equally to cases where the auxiliary data comes from an earlier time period. **c** An example of mobile phone interaction data. Each interaction contains the pseudonyms of the parties A and B, timestamp, type of interactions, direction (equal to "out" if A initiated it, "in" otherwise) and the duration for calls. In this example, the person identified by "gvuQjU" received a text from another person, identified by "dPefYb", to whom the former responded 2 minutes later. After 22 minutes, "gvuQjU" called another individual, identified by "LUrKAk", for a duration of 124 seconds. 29

3.2 **An example of a simplified 2-IIG**. We show a 2-IIG, $G^2_{i,\mathcal{T}}$ (left), with vertex set consisting of originating individual i (yellow) and its 1-hop (a, b, c and d, gray) and 2-hop (e, f, g, h and j, purple) neighbours, and edges between them displayed as arrows with thickness proportional to the number of interactions. The nodes marked with + (d, e and f) can be considered as out-of-network (i.e., $d, e, f \in \mathcal{I} \setminus \mathcal{I}''$), as they only have incoming edges in the 2-IIG. We further show the simplified 2-IIG $\bar{G}^2_{i,\mathcal{T}}$ (right), consisting of the originating individual i and the 1-hop neighbours, with one edge between any two nodes if there was at least one edge in 2-IIG. In the simplified 2-IIG, all nodes are endowed with behavioural features. 46

3.3 **One step of propagation to the neighbours on a simplified 2-IIG using graph attention. a** For each node in the 2-IIG, an embedding is computed at step $s+1$ using the embeddings at step s. The attention weights (denoted by α) are computed as shown in Alg. 1 (line 7), with superscript symbol s dropped here for illustration purposes. **b** The forward propagation step concatenates a node's embedding at step s to the weighted average of its neighbours' weights, to be given as input to an MLP layer followed by $\mathbb{L}_2$-normalisation. 47

3.4 **Probability of identification after one week.** For each $k \in \{1, 2, 3\}$, we plot p_k, the probability of identification within rank $R \in \{1, \ldots, 43,606\}$ when the time delay is $D = 1$ week, with the 95% confidence interval shown in light blue. **(Inset)** shows the probability of identification for ranks 1, 10 and 100, with error bars for the 95% confidence interval. 53

3.5 **Probability of identification when the time delay increases.** We plot p_k, the probability of identification within rank 1 for $k \in \{1, 2, 3\}$ when the time delay between the dataset and the adversary's auxiliary information is equal to D weeks. The auxiliary information is one week long. The 95% confidence interval is shown in light blue. The vertical grey lines correspond to holidays. The probability of identification decreases as $y = p_k(D = 1) - \alpha_k \times (D - 1)$, with $\alpha_1 = -0.006$, $\alpha_2 = -0.017$ and $\alpha_3 = -0.018$. 54

3.6 **Probability of identification for increasing time period length of auxiliary data.** For each $k \in \{1, 2, 3\}$, we plot p_k, the probability of correct identification ($R = 1$) when the adversary's auxiliary data $\mathcal{T}_2$ consist of L weeks, $1 \leq L \leq 20$ (the largest value for each k is marked). The 95% confidence interval is shown in light blue. **(Inset)** shows the difference quotient $\Delta p_k(L) = p_k(L) - p_k(L-1)$ for $2 \leq L \leq 20$. 56

3.7 **Attack's performance with increasing population size.** We show $p_k(N')$, the probability of identification within rank 1 for $k \in \{1, 2, 3\}$ in a population of size N'. The 95% confidence interval is shown in light blue. **(Inset)** shows the negative difference quotient $-\Delta p_k(N') = -(p_k(N') - p_k(N' - \Delta(N))/\Delta N'$. The probability of identification decreases with the population size N', but at increasingly slower rates. 57

3.8 **Fraction of identified people vs fraction of individual weeks.** For each $k \in \{1, 2, 3\}$, we plot the fraction of people that are identified in at least a given fraction of individual weeks, using our model (in blue) and according to a Poisson binomial distribution (in gray, averaged over 100 trials). Our attack identifies 38.4% (resp. 14.5% and 38.5%) of the people more often than expected for $k = 2$ (resp. for $k = 1$ and $k = 3$). 58

3.9 **Coefficients of a Logistic Regression for individual identifiability.** For each feature, we plot the coefficient c (with the 95% confidence interval) of a logistic regression classifier with whether a person is more or less identifiable than expected as dependent variable. Features are ordered decreasingly from top to bottom according to the absolute value of c. When holding all other features constant, these results suggest that having more interactions and a well-balanced interaction graph makes individuals more identifiable. 60

3.10 **Coefficients of a Logistic Regression for individual identifiability using each feature as sole predictor variable.** For each feature, we plot the coefficient c (with the 95% confidence interval) of a logistic regression classifier with whether a person is more or less identifiable than expected as target variable, and the feature as predictor variable. Features are ordered decreasingly from top to bottom according to the absolute value of c. 62

3.11 **Pairwise feature correlations.** For each pair of features, we compute the Pearson correlation coefficient, rounded to single decimal precision. 63

3.12 **Receiving Operator Characteristic (ROC) curves with Area Under the Curve (AUC) score for the node pair classification task for** $k = 2$. The performance of a random classifier is shown in solid gray. 65

3.13 **Receiving Operator Characteristic (ROC) curve with Area Under the Curve (AUC) score for the node pair binary classification task**. The performance of a random classifier is shown in solid gray. For each $k \in \{1, 2, 3\}$, our method outperforms the baselines. 67

3.14 p_k**, the probability of identification within rank** R **when the time gap is one week,** $R \in \{1, \ldots, N\}$**.** For each $k \in \{1, 2, 3\}$, our method outperforms all the other approaches. 68

3.15 **Probability of identification in a bluetooth close-proximity network.** We plot $p_{k=1}$, the probability of identification within rank R for $k = 1$. The 95% confidence interval is shown in light blue. Our method correctly identifies people $p_{k=1} = 26.4\%$ of the time based on their 1-IIGs. It is able to identify the right person $p_{k=1} = 60.1\%$ of the time among the top 10 candidates. 72

3.16 **Illustration of data splits in different evaluation scenarios**. We perform testing, validation and training on sets disjoint in the time periods (**a** and **b**), the identities of the originating individuals of the k-IIGs (**c**) or the time periods and the identities of the originating individuals of the k-IIGs (**d**). We use the green, blue and yellow splits for training, validation and testing, respectively. To compute validation and test metrics, we use the first week in the split as reference week and the second week as target week. Scenario **a** is used for the main experiment of the work, while the others are used to evaluate the generalisation ability of our attack. 75

3.17 **Analysis of attention weights.** We report the cumulative distribution function of the **a** normalised entropy and **b** normalised range of the attention weights used to aggregate the neighbours the $N = 43,606$ individuals. The results are computed over 2-IIGs from the reference week and aggregation steps $s = 1, 2$. . . 77

4.1 **Distribution of answers to two queries and their difference computed on datasets with $s = 0$ and $s = 1$.** In this example, the answers are retrieved from a simple QBS with Gaussian noise addition and rounding. 90

4.2 **Automated learning of a rule G.** We illustrate our approach to automatically combine the answers to a multiset of queries, via a rule G learned using a machine learning model. In our example, partial records r^u consist of values for attributes a_1 and a_2, while a_3 is the sensitive attribute. The adversary samples d auxiliary datasets (randomising the value of a_3 for the target record), then retrieves $d \times m$ answers to the queries (in red) from (differently seeded) QBSs protecting the datasets. The answers together with the target's sensitive attributes from the corresponding datasets are used to train a classifier G. 94

4.3 **Illustration of how QuerySnout explores the search space using evolutionary algorithms.** In this example, the search space consists of multisets of $m = 3$ queries relating to $n = 3$ attributes. Solutions with higher fitness are more likely to be selected. We use different colours to illustrate the changes to a query and black for unmodified query operators. 95

4.4 **Comparison between automated and manual attacks against SimpleQBS(τ, σ) on the Adult dataset.** We report the accuracy of attribute inference attacks discovered by QuerySnout (left) and of manual attacks tailored to each system (right) in the AUXILIARY (top) and EXACT-BUT-ONE (bottom) scenarios. We report the attack accuracy over 100 target records, averaged over 5 repetitions. 110

4.5 **Comparison between automated and manual attacks against SimpleQBS(τ, σ) on other datasets.** We report the attack accuracy over 100 target records, averaged over 5 repetitions. 111

4.6 **Impact of the number of queries on the accuracy of QuerySnout.** We report the accuracy of QuerySnout (which uses an evolutionary search), of a random search, and of a solution chosen uniformly at random as we vary the number of queries m. The results are computed on the Adult dataset in the AUXILIARY scenario. 114

4.7 **Impact of various search parameters on the accuracy of QuerySnout.** The results are computed on the Adult dataset in the AUXILIARY scenario. The vertical gray line indicates the default value used in the main experiment. . 115

4.8 **Impact of the dataset size on the performance of QuerySnout.** We show the accuracy of an attribute inference attack discovered by QuerySnout in the AUXILIARY scenario, for varying dataset sizes. The dotted vertical line represents the value 8000 which we use for experiments on Adult and Census. . 117

4.9 **Histogram of the number of difference queries in each solution found by QuerySnout in the AUXILIARY scenario.** Histograms are computed over all solutions from the 5 repetitions. 118

4.10 **Performance of attacks that use all the queries in the solution found by QuerySnout (blue), only the difference queries (orange), and the same number of random queries in the solution (green).** Histograms are computed over all solutions from the 5 repetitions. The vertical bars show the mean accuracy. 119

6.1 **Accuracy of the correlation inference attack on $n = 3$ variables and different target models.** We consider all the possible values of the correlation constraints $\rho(X_1, Y)$ and $\rho(X_2, Y)$ using a fine discretisation of the grid (200×200 cells). The colour of each cell corresponds to the accuracy of our correlation inference attack on synthetic datasets. The datasets are sampled using Gaussian copulas parametrised by correlation matrices that satisfy the constraints $\rho(X_1, Y)$ and $\rho(X_2, Y)$ defining the cell. 169

6.2 **Logistic Regression: Comparison between black-box and white-box correlation inference attacks.** We report results for white-box attacks using as features the model weights (B. 2) or the combined weights and confidence scores (B. 3). 171

6.3 **Multilayer perceptron (MLP): Comparison between black-box and white-box correlation inference attacks.** We report results for white-box attacks using as features the raw model weights (C. 2), the *canonical* weights (mapped to a canonical representation [131], C. 3), and the combined canonical weights and confidence scores (C. 4). 171

6.4 **Attack accuracy for different scenarios and number of variables in the dataset** n. The accuracy (with 95% confidence interval) is computed over 1000 target models. 173

6.5 **Impact of the constraints on the attack accuracy for different number of variables** n**.** We compare the results of our model-based and model-less attacks on the Logistic Regression and MLP models. 174

6.6 **Attack accuracy under scenario S2 and** $B = 5$ **correlation bins for different number of variables in the dataset** n. The accuracy (with 95% confidence interval) is computed over 1000 target models. 175

6.7 **Impact of mitigations on the accuracy of our attack.** We show the attack accuracy (with 95% confidence interval) over 1000 targets models for different number of variables n. 176

6.8 **Trade-off between the vulnerability of DP Logistic Regression models to our attack (left) and their utility (right).** We compute the metrics on 1000 models and report the 95% confidence intervals. 177

6.9 **Correlation constraint extraction from LR models for different number of queries** $\tilde{Q}$**.** We compute the $\mathbb{L}_1$ error between the inferred and the ground-truth values of the correlations: $\frac{1}{n}\sum_{i=1}^{n}|\hat{\rho}(X_i, Y) - \rho(X_i, Y)|$ (mean with 95% confidence interval).. 186

6.10 **Fifa19: Impact of the granularity of one-way marginals available to the attacker.** We report the accuracy (mean and standard deviation) of attacks on LR models as we vary the number of discretisation bins applied to the one-way marginals. 186

6.11 **Comparison between black-box and white-box CIAs against MLPs for different seed configurations between the shadow and target models**. We report the mean attack accuracy (with 95% confidence interval). 187

7.1 **Visualisation of the misalignment problem in shadow models trained by the classical adversary.** Due to symmetries in the architecture, the models learn similar features in different positions. Each row shows the activations maps of the first layer of a CNN, taking as input the aeroplane image (top centre). The first row shows activations computed on the target model $\mathcal{M}_T$; the other rows show activations computed on three shadow models trained on disjoint datasets and using a different seed. The coloured rectangles highlight similar activation maps across different models. 199

7.2 **Visualisation of the first layer of CNN shadow models** when changing only the seed of (b) the batch ordering, or (c) the dropout selection, or (d) the model initialisation with respect to the target model. Changing the seed of the model results in layer misalignment between the target model $\mathcal{M}_T$ and the shadow models. Changing the seeds controlling the other sources of randomness does not, although some of the feature maps are slightly different. 209

7.3 **Visualisation of the first layer of CNN shadow models** trained on a dataset that is (b) overlapping with or (c) disjoint from the training dataset of the target model. Changing the dataset slightly impact the activation maps but does not result in misalignment, even when using disjoint datasets. 210

7.4 **Visualisation of the first layer of CNN shadow models having the same weight initialisation as the target model.** The shadow models are trained using a different batch ordering, dropout selection, and a dataset disjoint from the target model. The activation maps are broadly aligned although we can see some differences, e.g., in columns 1 and 7. 210

7.5 **Visualisation of the first layer of CNN shadow models trained on datasets of different sizes**. Each dataset is a subset of the larger datasets. Increasing the size of the dataset gradually changes the activation maps. As the model becomes less overfitted and learns more robust features, the activation maps stabilise and remain aligned. 211

7.6 **ROC curves of attacks on the CIFAR10 dataset.** The white-box attacks use features extracted from the last layer (left) or from the last two layers (right). We also report results for the black-box attack using the output activations of the last layer. 225

7.7 **ROC curves of attacks on other datasets.** 225

8.1 **Number of potential duplicates detected for each distance d by the PDQ algorithm.** 253

8.2 **Examples of original and modified images for different hashing functions.** All four modified images evade detection for the threshold T while maintaining a small $\mathbb{L}_2$ perturbation per pixel and high visual similarity to the original image. 256

8.3 **Visual similarity between original and modified images.** We compute the cumulative distribution function (CDF) of the (a) $\mathbb{L}_2$ perturbation per pixel and (b) LPIPS distance for different algorithms and thresholds and all successfully attacked images over 10 experiments. A lower perturbation or distance indicates a higher visual similarity between the modified and original images. The perturbation increases slowly with the threshold, but remains small in all cases. 257

8.4 **False Positive Rate (FPR)** for different algorithms and thresholds. We use a database size of $N = 100$K and estimate the FPR by matching $M = 500$K images against the database. We show the mean with error bars for the standard deviation (which are very small) over 10 repetitions. 258

8.5 **Pairwise distances between hashes of the multiple modified images generated for the same image, for different algorithms and thresholds.** We generate $D = 50$ modified images for each ($N' = 100$) original image. 259

8.6 **Cumulative distribution function (CDF) of the $\mathbb{L}_2$ perturbations per pixel of the multiple perturbations generated for the same image for different algorithms and threshold.** The number of perturbations generated for each image is $D = 50$ and the number of images attacked is $N' = 100$. 259

8.7 **Optimality analysis of the black-box approach**. We compute the cumulative distribution function (CDF) for the $\mathbb{L}_2$ norm of perturbations using our black-box attack (BB-NES) against the DCT step and a threshold $T = 1.6$ (gray dotted line) for $N' = 1000$ images. The inset shows the attack effectiveness for the three attacks. 260

8.8 **Image distances across datasets.** Cumulative distribution function (CDF) for distances between the image hashes for different algorithms, number of images P and datasets. The CDFs are roughly stable for each algorithm for different number of images. 262

8.9 **False Positive Rate (FPR)** for each algorithm, threshold, and database size N. The number of images visually different from those in the database that are used to estimate the FPR is $M = 500$K. 263

8.10 **False Negative Rate (FNR)** for each algorithm, threshold, and database size N. The number of attacked images used to estimate the FNR is $N' = 1000$. . . 263

8.11 **Probability for PDQ to flag at least k images** shared by non-offenders (sharing 0 illegal images) and offenders (sharing 100 illegal images). Both offenders and non-offenders share a total of 1000 images. 266

8.12 **Probability for other hashing algorithms to flag at least k images** shared by non-offenders (sharing 0 illegal images) and offenders (sharing 100 illegal images). Both offenders and non-offenders share a total of 1000 images. 266

8.13 **First example of an original image with modifications obtained using our black-box attack for different perceptual hashing algorithms and thresholds.** We show the $\mathbb{L}_2$ perturbation per pixel. All the modified images are qualitatively very similar to the original image. 270

8.14 **Second example of an original image with modifications obtained using our black-box attack for different perceptual hashing algorithms and thresholds.** We show the $\mathbb{L}_2$ perturbation per pixel. All the modified images are qualitatively very similar to the original image. 271

8.15 **Third example of an original image with modifications obtained using our black-box attack for different perceptual hashing algorithms and thresholds.** We show the $\mathbb{L}_2$ perturbation per pixel. All the modified images are qualitatively very similar to the original image. 272

8.16 **Fourth example of an original image with modifications obtained using our black-box attack for different perceptual hashing algorithms and thresholds.** We show the $\mathbb{L}_2$ perturbation per pixel. All the modified images are qualitatively very similar to the original image. 273

8.17 **Fifth example of an original image with modifications obtained using our black-box attack for different perceptual hashing algorithms and thresholds.** We show the $\mathbb{L}_2$ perturbation per pixel. All the modified images are qualitatively very similar to the original image. 274

Chapter 1

Introduction

1.1 Motivation

Technology has become ubiquitous today as it assists us in our day-to-day lives. We use our smartphones to connect with our friends and family, read the latest news, shop online, and pay bills. We use the Internet to search for a job, explore travel destinations, watch movies and sports, play games, discover communities that share our hobbies, and follow online courses. During the recent COVID-19 pandemic, online technology became the main way to socialise for both work and leisure. Even offline, smart devices such as smart speakers, smart cars, smart meters, and smart watches fulfilling different needs have proliferated.

Many of these technologies collect, process, and share sensitive data with different entities. Examples of sensitive data collection include: every time we click on a website, select an item or even scroll over a webpage, a log of our interaction is collected; communication providers, websites, and smartphone apps routinely collecting our location; and mobile network operators collecting call detail records which document who we talk to, when, and for how long. In many cases, this data is then shared with other parties, e.g. national statistical agencies release microdata and statistics about their population [246, 273, 335]; big tech platforms with access to locations of millions of individuals released mobility reports [14] and density maps [152] during the recent COVID-19 pandemic; and companies sell access to clinical data [330]. The

drive to collect and share data is further intensified by the unprecedented advances in artificial intelligence of the last few years, with machine learning (ML) models being used to automate or perform new tasks. Indeed, such models perform better as they use more training data [317, 174], leading to ever larger datasets being collected to train these models [291].

Individuals' right to privacy is enshrined as a fundamental human right in many constitutions around the world and the Universal Declaration of Human Rights (Article 12) [22]. Privacy means that individuals have the right to be let alone and choose what information to disclose, to whom, and for what purpose. This right is required for us to feel safe, develop trust, shape our identity, and make independent decisions without being influenced by others. In spite of privacy being a fundamental human right, the scale and pervasiveness of data collection, processing, and sharing seriously challenges it. Today's digital landscape is dominated by big tech companies, whose business model – dubbed as "surveillance capitalism" [371] – rests on collecting behavioural data and using it to predict our behaviour for targeted advertising. The awareness that our actions may be monitored and logged, and potentially used to influence our behaviour, restricts our freedom. Even more explicitly, our data can be surreptitiously used for highly personalised political advertising, as demonstrated by the Facebook-Cambridge Analytica scandal [154, 49, 234]. Another example of data being weaponised against an individual is a Catholic priest resigning after commercially available data from the Grindr app was linked to his identity by a Catholic news site who claimed he went to gay bars [43].

Understanding how modern technologies affect our privacy, at every stage of the process, is a key question. Furthermore, doing this before technologies are deployed is essential. For instance, the datasets used to train machine learning models on are often sensitive, meaning that as models are being shared widely through APIs or on devices, their release raises concerns about the potential leakage of sensitive information about their training data. Another timely example of sensitive data processing is perceptual hashing-based client-side scanning (PH-CSS), a system developed for detecting child sexual abuse material (CSAM) in end-to-end-encrypted messaging platforms [76, 17]. PH-CSS scans images on device for illegal content and sends them unencrypted for manual inspection if a close match is found. Since images flagged as CSAM would be shared in raw (unencrypted) form with the platform for a manual review, a concern is

that people's private images could be erroneously flagged, leading to privacy violations [9, 260].

Privacy is typically evaluated through the amount of sensitive information that untrusted parties, referred to as adversaries, can learn from a data release. Evaluating whether a system to collect, process or share data preserves the privacy of individuals naturally involves defining who an adversary might be, what they are trying to learn about an individual, and what information they may already have about the individual and the system. This adversarial approach to privacy is instrumental to some privacy legislations in the case of data sharing. The Recital 26 of the European Union's General Data Protection Regulation (EU GDPR) [2], defines anonymous data as "information which does not relate to an identified or identifiable natural person or to personal data rendered anonymous in such a manner that the data subject is not or no longer identifiable." As this limits the harm to individuals resulting from the processing of their data, anonymous data does not fall under the scope of data protection, meaning that it can be shared to third parties and transferred abroad without the consent of the data subjects. Recital 26 further states that to "ascertain whether means are reasonably likely to be used to identify the natural person, account should be taken of all objective factors, such as the costs of and the amount of time required for identification, taking into consideration the available technology at the time of the processing and technological developments". This can be understood as a requirement for the data collector to both implement anonymisation techniques and evaluate the risk of re-identification. This subsumes a need to consider the state-of-the-art attacks of the day and to continually re-evaluate whether the data remains anonymous [19].

The adversarial approach to privacy has exposed numerous privacy vulnerabilities and flaws of modern technologies such as anonymisation techniques [320, 227, 228, 90, 92, 278, 99, 43] and machine learning models [126, 302, 283, 205, 54, 150], improving our understanding of their limitations. It has also raised the standard of what makes a technology robust to attacks and what constitutes anonymised data. However, this process is contextual and labour-intensive: attacks are system-specific and rely on specific assumptions, often making them inapplicable to different settings than the ones they were originally developed for. More fundamentally, existing attacks only provide a lower bound of the risk, as new technical advances may allow to improve a given attack method, or new auxiliary information becomes available allowing to

re-identify datasets which may have previously been considered to be safe [262].

A complementary approach is to pre-empt attacks by ensuring that a system provably prevents privacy attacks with mathematical guarantees. A great amount of research has been dedicated to developing formal privacy definitions [101, 107, 87, 98, 60]. Today, the most widely accepted definition for releasing information while protecting the privacy of individuals is differential privacy (DP) [107]. DP requires that the output of a computation, such as the average income of UK citizens, should be the same (in a statistical sense) whether or not any individual's data is included in the computation. This definition has many appealing properties [107]. In particular, it protects against very strong adversaries, bounding the leakage of information about an individual even by a "worst-case" adversary that knows the entire dataset except for the individual [107]. However, DP is not yet widely adopted, one of the reasons being that the design of mechanisms satisfying the definition with both strong privacy guarantees and good utility is often challenging [264, 166, 312]. Consequently, real-world systems often use either ad-hoc anonymisation techniques [246, 123], which are not future-proof as they can be thwarted by new attacks [129, 265] or use lower privacy guarantees that lead to better utility but do not provide formal protection against the strongest attackers [279, 335]. Furthermore, the formal guarantees of DP can be difficult to convey to non-experts [82] and its implementations are prone to errors [312, 159]. Complementing formal guarantees with concrete measures of the risk in practice, e.g., when worst-case upper bounds are not helpful, requires, again, developing state-of-the-art attacks.

In this thesis, we argue that in spite of all this research, there is a need to develop new threat models and methodologies to comprehensively evaluate the privacy risks of modern data processing technologies in practice. We perform five case studies on four different technologies and five different contexts in which data is shared, developing new threat models and methods to quantify privacy risks. More specifically, we contribute new attacks against: anonymisation techniques for interaction datasets relying on pseudonymisation and frequent re-pseudonymisation (Chapter 3), query-based systems providing answers to queries about a dataset (Chapter 4), machine learning models (Chapters 6 and 7), and perceptual hashing-based client-side scanning systems (Chapter 8).

1.2 Contributions

In this thesis, we make the following contributions.

Our first contribution relates to the most straightforward form of data sharing: providing access to a dataset of records. Re-identification attacks have long been used to identify individuals in pseudonymised datasets using matching auxiliary information, calling into question their anonymity [320, 227, 228, 90, 92, 70, 99]. However, such attacks rely on the assumption that auxiliary information is fairly stable over time (gender, zip code, etc.) or from the same time period as the dataset (spatio-temporal points, movies watched, etc.). This assumption limits their applicability to dynamic datasets such as interaction data collected by mobile phone carriers and contract tracing apps. For instance, a data controller could frequently change the pseudonyms of individuals, e.g., weekly, to limit the risk of re-identification. This mitigation strategy, known as re-pseudonymisation, reduces both the amount of auxiliary knowledge that could be used to perform an attack and the amount of information recovered if the attack succeeds. Another motivating example is an individual changing their mobile phone number to hide their identity, e.g., from law enforcement agencies: can records of the individual's interactions collected from different phone numbers be pieced together?

In **Chapter 3**, we demonstrate that people's interaction behaviour is stable over multiple weeks and can be used to identify individuals in large-scale pseudonymous datasets. We propose a graph neural network embedding model that extracts a profile of an individual which can be used to identify them across different time periods. In a mobile phone metadata dataset of more than 40K people, our model correctly identifies 52% of individuals based on their 2-hop interaction graph. We further show that the profiles learned by our model are stable over time and that 24% of people are still identifiable after 20 weeks. Our results suggest that people with well-balanced interaction graphs are more identifiable. We then apply our attack to Bluetooth close-proximity networks and show that even 1-hop interaction graphs are enough to identify people more than 26% of the time. Our results provide strong evidence that disconnected and even re-pseudonymised interaction data can be linked together, making them personal data

under the EU's GDPR.

Due to multiple high-profile successful re-identifications of public datasets [320, 30, 227] and to studies such as ours which quantify the risk of identification in different types of data [183, 90, 92, 99, 334, 116, 81, 214], it is today widely accepted that anonymising datasets is, in general, very difficult [245]. A promising alternative to sharing individual-level records is to instead publish aggregate statistics computed over many records. Aggregate statistics can be either pre-defined (e.g., census tables) or answers to queries from a query-based system (QBS, e.g., SQL queries to a database). An important question is whether aggregation is sufficient to prevent adversaries from learning sensitive information about individuals in the dataset; researchers have shown that this is not necessarily the case [97, 96, 101, 264, 129]. In particular, the state-of-the-art attack against pre-defined statistics uses a machine learning classifier to infer the membership (presence or absence) of a record in the dataset from which the statistics are extracted [264]. The classifier is trained on statistics extracted from auxiliary datasets in which the record is present half of the time. Evaluating the privacy provided by QBSs presents an additional layer of complexity: it requires evaluating all possible sets of queries of a given size within a specific syntax, which is prohibitively expensive. Hence, attacks on QBSs typically rely on manually selected sets of queries whose answers can be combined to reveal sensitive information [67, 277, 265, 129]. Such attacks however require time and expertise to develop, need to be manually tailored to the specific systems attacked, and are limited in scope.

In **Chapter 4**, we focus on a type of vulnerability called attribute inference attack, whose goal is to infer a target record's sensitive attribute from the answers to a set of queries. We develop QuerySnout, the first method to automatically explore the space of queries within a well-defined syntax in order to find a set of queries susceptible to an attack. We use evolutionary search techniques to explore the space of queries and a machine learning classifier to infer the sensitive attribute from answers to the queries selected. We perform an extensive evaluation of our method, instantiating it on three real-world datasets, a variety of protection mechanisms, and with different levels of adversary knowledge about the dataset. We show the attacks found

by QuerySnout to consistently equate or outperform, sometimes by a large margin, the best attacks from the literature. We finally show how QuerySnout can be extended to QBSs that require a budget, and apply it to a simple QBS with DP guarantees. Taken together, our results show how powerful and accurate attacks against QBSs can be found by an automated system, allowing for highly complex QBSs to be automatically tested "at the pressing of a button".

While we have so far focused on machine learning (ML) as a tool to learn attacks, we now turn to it as the subject of our study. ML models are often trained on sensitive and proprietary datasets. They are another instance of the *aggregation* framework, as they are the result of applying a training procedure to a specific dataset. However, unlike aggregate statistics, models are black boxes, in the sense that manually characterising what a model "remembers" about its training dataset is difficult, if not impossible. Indeed, learning from a dataset typically involves performing a complex, iterative, and stochastic procedure, over random subsets of the dataset (so-called mini-batches) to update an often very large number of parameters using gradient descent. This makes the relationship between a model and the dataset it was trained on largely untraceable. Empirical attacks, in the form of inference games, have instead been used to analyse, post-hoc, whether a trained model "remembers" a specific piece of sensitive information (a secret) [127, 23, 302, 54]. Shadow modelling is the state-of-the-art technique used in inference attacks [23, 302, 52]. It is conceptually similar to the one we used against aggregate statistics [264, 265], as it involves training a classifier on features extracted from shadow models trained by the adversary on auxiliary datasets having different values of the secret. A key difference is that the classifier can only be trained on features extractable from the model under the assumed threat model, e.g., only model confidences are accessible to a black-box adversary [302].

So far, inference attacks have mostly focused on inferring the membership of a given record in the dataset, something called a membership inference attack (MIA) [302, 283, 229, 332, 68, 52]. MIAs have received great attention because they are conceptually simple and generally applicable to any model, dataset and type of adversary access to the model. Another reason is to test the guarantees of DP, which formally provides resistance to MIAs. Membership is,

however, only one of the characteristics of the training dataset that can be remembered by a model. In particular, there has been comparatively little work studying whether models encode dataset-level characteristics [131, 365, 318]. This is a non-trivial question. On the one hand, the ideal model should learn nothing more than the intended probability of the output label given an input sample $\mathcal{M}(x) \simeq \Pr(Y|X = x)$. However, models are trained to minimise a loss function computed over a specific finite dataset of samples from the underlying dataset distribution $\Pr(X, Y)$ [140]. The dataset distribution $\Pr(X, Y)$ and, by extension, $\Pr(X)$, may very well end up influencing the final model in practice.

In **Chapter 6**, we study a new type of leakage in ML models: the leakage of correlations between input variables of a tabular training dataset. We argue that correlations can be sensitive in and of themselves and that they can be used as building blocks for stronger attribute inference attacks, in particular enabling weak adversaries that do not know the training dataset distribution. The leakage of correlations could lead to privacy violations, as correlations are not theoretically protected by differential privacy. We here first propose a model-less attack, showing how an adversary can exploit the spherical parameterisation of correlation matrices to make an informed guess based on the correlations between the input variables and the target variable alone. Second, we propose a model-based attack, showing how an adversary can exploit black-box access to the model to infer the correlations using shadow models trained on synthetic datasets. Third, we evaluate our attacks against Logistic Regression and Multilayer Perceptron models on both synthetic and real-world tabular datasets and show that the models indeed leak correlations. We finally show how the dataset correlations extracted using our attack can be used as building blocks for stronger attribute inference attacks.

ML privacy risks, and in particular membership inference attacks, have mostly been studied in the black-box setting, where the adversary accesses the model through an API [302, 283, 68, 201, 52]. Recently, white-box adversary access (i.e., access to the model weights) is becoming a concern in a range of contexts including on-device models (available in apps and edge devices) and open-source models. White-box access to model weights increases the attack surface compared to the black-box setting which only provides query access to a model, and should thus

lead to more powerful attacks. In spite of this, directly extending the shadow modelling technique from the black-box to the white-box setting has been shown, in general, not to perform better than black-box MIAs [229, 131, 365, 80].

In **Chapter 7**, we study misalignment, a known characteristic of deep neural networks stemming from their weight symmetries [151, 21], as a key reason for the poor performance of white-box MIAs. In the shadow modelling context, misalignment means that, while the shadow models learn similar features in each layer, the features are located in different positions [131]. We present the first systematic analysis of the causes of misalignment in shadow models and show the use of a different weight initialisation to be the main cause of shadow model misalignment. We extend several re-alignment techniques, previously developed in the model fusion literature, to the shadow modelling context, where the goal is to re-align the layers of a shadow model to those of the target model. We show re-alignment techniques to significantly reduce the measured misalignment between the target and shadow models. Finally, we perform a comprehensive evaluation of white-box membership inference attacks (MIA). Our analysis reveals that (1) MIAs suffer from misalignment between shadow models, but that (2) re-aligning the shadow models improves, sometimes significantly, MIA performance.

A common theme in the contributions presented in chapters 4, 6, and 7 is that conceptually, the methods used to evaluate the privacy of QBSs and ML models are very similar. Both involve training a classifier on instances of the aggregate (query answers computed on shadow QBSs and shadow models, respectively) generated by an adversary to have different values of the secret. In both cases, determining which features to extract so as to maximise attack accuracy and, thus, obtain a tight estimate of the risk is a highly complex task. However, the reasons why this is the case are different. For QBSs, the key challenge is that in real-world systems only a limited number of queries can be asked, and the number of possible query sets of a given size is prohibitively large even in a limited syntax. For models, the key challenges are the high-dimensionality of features and, in the case of deep neural networks, the existence of symmetries which lead to feature misalignment and further complicate the classifier's task. Our technical contributions specifically tackle these challenges.

Our work so far has mostly focused on evaluating the privacy protection of intentional data releases, i.e., that are deemed useful in some way to the entities accessing them. For instance, statistics are used to inform decisions and models are used to make predictions. We now apply the adversarial approach to a different kind of data processing technology with privacy implications, the very recently proposed perceptual hashing-based client-side scanning (PH-CSS). PH-CSS is seen as a promising solution by tech companies, governments and researchers to detect child sexual abuse material (CSAM) in end-to-end encrypted (E2EE) communications. PH-CSS computes a fingerprint of every image shared by a user using a perceptual hashing algorithm and compares it against a database of fingerprints of known CSAM images. If the fingerprint of an image is closer to any fingerprint of images in the database by a quantity lower than a detection threshold, that image is flagged. Flagged images would be shared raw (unencrypted) with relevant entities for a manual inspection. As detection would take place directly on the user devices before encryption, its proponents claim it maintains the privacy of users. The goal of the system is indeed to only flag (edited copies of) CSAM images in the database and nothing else. But significant privacy concerns have been raised about the system's potential to falsely flag non-CSAM images [9], as sharing such images defeats the purpose of E2EE and could lead to privacy violations. Another concern relates more generally to the immense scope of PH-CSS deployment, as it would be installed on people's personal devices and scan all images shared by users of E2EE messaging platforms. A key question is thus whether the privacy and security risks of PH-CSS, such as the risk of false positives, can be balanced by effective detection [260]. While theoretically the number of falsely flagged non-CSAM images can be controlled by reducing the detection threshold, doing so may open the door to imperceptible adversarial manipulations of CSAM images that allow them to evade detection by the PH-CSS system.

In **Chapter 8**, we study whether PH-CSS is robust under adversarial settings, proposing three detection avoidance attacks – a general black-box attack and two white-box attacks for discrete cosine transform-based algorithms – against perceptual hashing algorithms. In a large-scale evaluation, we show PH-CSS to be highly vulnerable to detection avoidance attacks in a black-box setting, as we are able to successfully modify more than 99.9% of images while preserving

the content of the image. We furthermore show our attack to generate diverse perturbations, suggesting that straightforward mitigation strategies may be ineffective. Finally, we show that the larger detection thresholds necessary to make the attack harder would probably require more than one billion images to be flagged and decrypted daily, raising strong privacy concerns. Taken together, our results shed serious doubts on the robustness of PH-CSS solutions currently proposed by governments, organisations, and researchers around the world.

1.3 Thesis outline

The remainder of this thesis is organised as follows.

Chapter 2 presents key concepts used throughout the thesis.

Chapter 3 presents an identification attack against pseudonymised interaction data using auxiliary information from a disjoint time period.

Chapter 4 presents a method for automating the discovery of attribute inference attacks against query-based systems providing answers to counting queries.

Chapter 5 provides a background on machine learning (ML) privacy, describing a unified threat model for inference attacks against ML models, reviewing the literature on the topic, and identifying open questions.

Chapter 6 presents a new type of leakage in ML models, the leakage of correlations between input variables of a tabular training dataset.

Chapter 7 presents a white-box membership inference attack against ML models based on re-aligned shadow models.

Chapter 8 presents black-box and white-box detection avoidance attacks against perceptual hashing-based client-side scanning.

Chapter 9 concludes this thesis by summarising our contributions and presenting our perspective on promising avenues for future work.

1.4 Publications

The work presented in this thesis comes from the following peer-reviewed articles (the asterisk* denotes the first author(s) of a paper):

- Creţu, A.-M.*, Monti, F., Marrone, S., Dong, X., Bronstein, M., and de Montjoye, Y.-A. (2022). "Interaction data are identifiable even across long periods of time." Nature Communications, 13(1), 313.
- Jain, S.*, Creţu, A. M.*, and de Montjoye, Y. A. (2022). "Adversarial Detection Avoidance Attacks: Evaluating the robustness of perceptual hashing-based client-side scanning". In 31st USENIX Security Symposium (USENIX Security 22) (pp. 2317-2334).
- Cretu, A.-M.*, Houssiau, F.*, Cully, A., and de Montjoye, Y.-A. (2022). "QuerySnout: Automating the Discovery of Attribute Inference Attacks against Query-Based Systems". In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (pp. 623-637).

The thesis also includes work based on the following, currently non peer-reviewed manuscripts:

- Cretu, A.-M.*, Guepin, F.*, and de Montjoye, Y.-A. (2022). "Correlation Inference Attacks against Machine Learning Models". A version of this manuscript is available at `https://arxiv.org/abs/2112.08806`.
- Cretu, A.-M.*, Jones, D., de Montjoye, Y.-A., and Tople, S. (2023). "Re-aligning Shadow Models can Improve White-box Membership Inference Attacks". A version of this manuscript is available at `https://arxiv.org/abs/2306.05093`.
- Gadotti, A.*, Rocher, L., Houssiau, F., Cretu, A.-M., and de Montjoye, Y.-A. (2023). "Anonymization: The imperfect science of using data while preserving privacy".

Chapter 2

Preliminaries

In this chapter, we introduce several key concepts used throughout the thesis. In particular, we formally define: record and dataset (used throughout the thesis), pseudonymisation and re-pseudonymisation (used in chapter 3), aggregate (a unified abstraction of query-based systems and machine learning models which we focus on in chapters 4- 7), and differential privacy (used in chapters 4- 6).

2.1 Individual-level data

2.1.1 Definition

Definition 2.1 (Record)**.** *A record represents the data x collected about one individual $u \in \mathcal{U}$, with $\mathcal{U}$ denoting a population of individuals or users of a service.*

Definition 2.2 (Dataset)**.** *A dataset D is a multiset of records collected about a set of users $U \subset \mathcal{U}$, such that no two records relate to the same user. A dataset can contain several identical records as long as they relate to different individuals.*

When the data can be represented as values for a set of attributes $\mathcal{A} = \{a_1, ..., a_n\}$, we will write $x = (x_1, x_2, \ldots, x_n)$, with $x_i \in \mathcal{V}_{a_i}$ and $\mathcal{V}_{a_i}$ denoting the set of admissible values for attribute

a_i. Such data is referred to as structured or tabular data. Data can also be unstructured, e.g., in the image, text or graph domain, meaning that it does not come in a structured, pre-defined format. In this thesis, we study both types of data: tabular data (chapters 4, 6 and 7) and unstructured data from the image (chapters 7 and 8) and graph domain (chapter 3). We refer the reader to the corresponding chapters for precise definitions and examples.

Data collected from an individual u is usually labeled with a direct identifier ID(u), e.g., name or phone number, which links the record to a natural person. In our formalism, this can be modelled as an additional attribute.

2.1.2 Pseudonymisation

Releasing a dataset of records with direct identifiers leads to direct re-identification, i.e., linking of a record with the individual from whom it was collected. Re-identification poses a privacy risk and can lead to harm against the individual when the record, or part of it, holds sensitive information[1], such as a medical diagnosis. Note that what counts as sensitive information is context-dependent: even seemingly innocuous information such as movie ratings can be sensitive if it can be correlated with religion or political preferences [227]. In some contexts, participation to a dataset can be sensitive, such as genomic datasets associated with a sensitive trait or datasets collected by dating apps [155, 43].

Pseudonymisation aims to reduce the risk of re-identification by removing direct identifiers and replacing them with new identifiers.

Definition 2.3 (Pseudonymisation)**.** *Let D be a dataset collected from a set of users U, with each record labeled with a direct identifier ID(u). Pseudonymisation refers to defining new identifiers for users, i.e., a new mapping ID' from the set of users U to e.g., strings, such that (1) identifiers are unique ID'(u) $\neq$ ID'(v), $\forall u, v \in U$ and (2) it is not possible to determine the identity of a user u based solely on ID'(u). Typically, these new identifiers are selected randomly.*

[1] We use the term "sensitive" in its colloquial sense, to also include private information that might be compromising about an individual even though it is not special category data in the EU GDPR's Art. 9 sense.

Robust (computationally infeasible to revert) pseudonymisation can be achieved, for instance, by applying a cryptographic hash function to the concatenation between the direct identifier and a secret salt (i.e., a long, randomly generated sequence of characters). Pseudonymisation can prevent re-identification via direct identifiers if correctly implemented.[2] However, it may still be possible to re-identify an individual indirectly, by matching attributes in their record with public or otherwise available information such as gender and date of birth [320]. Such attributes are commonly referred to as *quasi-identifiers* [321].

Definition 2.4 (Quasi-identifier)**.** *A quasi-identifier is a subset of attributes which are not direct identifiers and, together, can uniquely or almost uniquely identify an individual.*

The definition of quasi-identifiers is context-dependent, as quasi-identifiers are typically assumed to be "knowable" by an adversary.

The key problem with pseudonymisation is that people are very unique: only a few quasi-identifiers are sufficient to uniquely identify them, even in large populations. In 1997, Sweeney re-identified the medical record of William Weld, the governor of Massachusetts, in a pseudonymised public dataset based on his zipcode, birth date, and gender. She further estimated that 87% of Americans can be uniquely identified based on these attributes according to the 1990 census [320]. In a seminal paper, Narayanan and Shmatikov examined a dataset of movie ratings of a large fraction of Netflix users released by the company to improve its movie recommendation algorithm [227]. They showed that with 8 movie ratings and their approximate dates, 99% of records can be uniquely identified. In 2013, 4 spatio-temporal points (antenna and hourly time) were shown to be enough to uniquely identify 95% of individuals in a mobile phone dataset of 1.5M individuals [90]. Similar results have been found for health data [321, 356], shopping data [92], browsing data [99], apps installed on a smartphone [10], smart meter records [342], and cryptocurrency transactions [138]. To address the unicity issue, Sweeney formalised a notion of dataset privacy, k-anonymity, requiring that every record should be indistinguishable from at least $k-1$ others [320]. However, k-anonymity is not a privacy panacea, as it still allows an adversary to infer sensitive information about individuals when certain conditions

[2]Datasets have been incorrectly (reversibly) pseudonymised in the past [327, 190].

about the dataset are met [208, 198, 71]. More fundamentally, the transformations to the data required to achieve k-anonymity degrade its utility, and this degradation is even more severe as more data is collected about a person. In particular, high-dimensional sparse datasets are notoriously difficult to anonymise properly while maintaining utility [360, 90, 99].

2.1.3 Re-pseudonymisation

An alternative approach for dynamic datasets collected on a regular basis, such as mobile phone metadata or smart meter data, is to frequently re-pseudonymise data, meaning that users' identifiers are replaced over time (e.g., on a weekly basis) with new ones.

Definition 2.5 (Re-pseudonymisation)**.** *Let $\mathcal{T}_1, \ldots, \mathcal{T}_k$ denote k disjoint time periods (e.g., k consecutive weeks) and U a set of users. Assume that for every time period i, a record $x_{\mathcal{T}_i}(u)$ is collected about a user u. Re-pseudonymisation means associating different pseudonyms to the same user in each of the time periods: $ID_{\mathcal{T}_1}(u), \ldots, ID_{\mathcal{T}_k}(u)$ such that $ID_{\mathcal{T}_i}(u) \neq ID_{\mathcal{T}_j}(u), \forall 1 \leq i < j \leq k$. By contrast, pseudonymisation would have meant associated a pseudonym $ID_{\mathcal{T}}(u)$ to all the records $\{x_{\mathcal{T}_i}(u), i = 1, \ldots, k\}$, with $\mathcal{T} = \cup_{i=1}^{k} \mathcal{T}_i$.*

Re-pseudonymisation can be achieved, for instance, by generating pseudonyms in the first time period and shuffling them in every subsequent time period or by generating pseudonyms in each time period using a different salt. Re-pseudonymisation aims to reduce the risk of re-identification by reducing both (1) the amount of auxiliary information that could be used to perform an attack (i.e., an adversary can only use data from within the same time period) and (2) the amount of information recovered if the attack succeeds. Re-pseudonymisation has been proposed as a privacy-preserving method for collecting smart meter data (see Tudor et al. for an overview [334]), Bluetooth signals in contact tracing apps [240], and vehicle locations [217], and for sharing mobile phone metadata [94].

The length of the time period over which a pseudonym is kept constant depends on the context, i.e., on the intended use case of the data but also on its sensitivity. For instance, longer time periods can enable longitudinal analyses of individuals' data, but they increase the risk of

re-identification as they increase the amount of auxiliary information (quasi-identifiers) that could be used by an adversary. In chapter 3, we study the extent to which mobile phone and Bluetooth interaction data are identifiable over time, i.e., by an adversary using auxiliary information from a disjoint time period. One of the key applications of our method is to piece together weekly re-pseudonymised interaction data.

2.2 Aggregate data

Due to the difficulty of anonymising individual-level data, researchers have proposed to move away from sharing individual-level data [245] and instead to only publish aggregate statistics [89]. We here formally introduce the concept of aggregate, which we argue can be used to also represent other forms of data releases such as machine learning models. We use this concept in chapters 4- 7. We also introduce the concept of differential privacy [107] which is used to enforce privacy of aggregates. We use this concept in chapters 4- 6.

2.2.1 Definition

Definition 2.6 (Randomised mechanism)**.** *: Let $\mathcal{D}$ be the set of all possible datasets of a certain type. A randomised mechanism $\mathcal{M} : \mathcal{D} \longrightarrow \mathcal{Y}$ is a function that given as input a dataset D outputs a random variable $\mathcal{M}(D)$ taking values in $\mathcal{Y}$.*

Example 1 (Counting query)**.** *The function that returns the number of records in a dataset: $\mathcal{M} : \mathcal{D} \longrightarrow [0, \infty)$, $\mathcal{M}(D) = |D|$ is a randomised mechanism. In this example, $\mathcal{M}(D)$ is a deterministic random variable.*

Example 2 (Counting query perturbed with Gaussian noise)**.** *The function that returns the perturbed number of records in a dataset: $\mathcal{M} : \mathcal{D} \longrightarrow \mathbb{R}$, $\mathcal{M}(D) = |D| + N$, with $N \sim \mathcal{N}(0, 1)$ a sample of a standard normal distribution, is a randomised mechanism.*

Randomised mechanisms need not output one scalar; their outputs can also be multi-dimensional.

Example 3 (Contingency table of a dataset perturbed with Gaussian noise)**.** *Let $\mathcal{D}$ be the set of all possible datasets of records of n attributes $a_1, \ldots, a_n$ taking values in $\mathcal{V}_{a_1}, \ldots, \mathcal{V}_{a_n}$ respectively. The mechanism $\mathcal{M} : \mathcal{D} \longrightarrow \mathbb{R}^{|\mathcal{V}_{a_1}| \times \ldots \times |\mathcal{V}_{a_n}|}$, defined as :*

$$\mathcal{M}(D) = \big(T(D, v_1, \ldots, v_n) + n_{v_1,\ldots,v_n}\big)_{(v_1,\ldots,v_n) \in \mathcal{V}_1 \times \ldots \times \mathcal{V}_n}$$

where $T(D, v_1, \ldots, v_n)$ denotes the number of records in the dataset D having v_i as value for the i-th attribute for $i = 1, \ldots, n$ and $n_{v_1,\ldots,v_n} \sim \mathcal{N}(0, 1)$ is a sample from the standard normal distribution.

An important example of a randomised algorithm is any learning procedure for fitting a model, i.e., for optimising its parameters, applied to a specific training dataset D.

Example 4 (Machine learning model)**.** *Let $\mathcal{T}$ be a learning (training) algorithm for a machine learning model consisting of all the steps taken on a training dataset D to produce a trained model. The training algorithm is a randomised mechanism.*

A training algorithm $\mathcal{T}$ is usually an optimisation algorithm, consisting of (1) a model, e.g., architecture loss function, and hyperparameters such as the learning rate or batch size (for mini-batch gradient descent) which define a set of parameters and (2) a procedure, e.g., mini-batch gradient descent to optimise the parameters of the model. In many cases, learning algorithms are non-deterministic, as they contain one or more sources of randomness. For instance, fitting a random forest involves sampling subsets of features for each tree and node, while training a neural network involves random initialisation of the model's parameters, random sampling of mini-batches, and in some cases a randomised regularisation component e.g., dropout [310].

Definition 2.7 (Aggregate)**.** *Let $\mathcal{M} : \mathcal{D} \longrightarrow \mathcal{Y}$ be a randomised mechanism. Given a dataset D, an aggregate of D is a sample of the random variable $\mathcal{M}(D)$.*

Our definitions of randomised mechanism and aggregate make the clear distinction that the former is a (possibly randomised) computation and the latter is the result of a computation. Examples of aggregates include: aggregate statistics extracted from a dataset, trained machine learning models, and generative models fitted on a dataset.

2.2.2 Differential privacy

Differential privacy (DP) [107, 108] is a privacy definition requiring that the output of a computation over a dataset of records should not depend too much (statistically) on the presence or absence of any one record, as controlled by a parameter ϵ:

Definition 2.8 ((ε, δ)-Differential Privacy)**.** *Let $\varepsilon > 0$ and $\delta \in [0, 1)$. A randomised mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{Y}$ is said to satisfy (ε, δ)-differential privacy if for every pair of neighboring datasets D and $D' \in \mathcal{D}$ (i.e., differing in exactly one record) and for any subset of outputs $Y \subset \mathcal{Y}$, the following holds:*

$$Pr(\mathcal{M}(D) \in Y) \leq e^{\varepsilon} Pr(\mathcal{M}(D') \in Y) + \delta$$

where the randomness is taken over the coin flips of $\mathcal{M}$.

If $\delta = 0$, $\mathcal{M}$ is said to be ε-differentially private. The parameter ε is commonly referred to as the "privacy budget". The lower its value, the stronger the privacy of the mechanism, as the distributions of $\mathcal{M}(D)$ and $\mathcal{M}(D')$ become harder to distinguish. The parameter $\delta > 0$ allows for a relaxation of the stronger definition that uses $\delta = 0$ [107]. There are many situations where allowing $\delta > 0$ enables the construction of mechanisms having better utility [8].

A key interpretation of differential privacy is that it protects against very strong adversaries. Indeed, even a "worst-case" adversary that knows the entire dataset D except for the membership of one record x is unable to distinguish between the two distributions $\mathcal{M}(D)$ and $\mathcal{M}(D \cup \{x\})$.

Differential privacy has two important properties that we will use in this work: composition and post-processing.

Proposition 1 (Composition [108])**.** *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two randomised mechanisms that are ε_1 and ε_2-differentially private, respectively. Then the composed mechanism $(\mathcal{M}_1, \mathcal{M}_2)$ is $(\epsilon_1 + \epsilon_2)$-differentially private.*

For instance, if an analyst retrieves the answer to two different queries, each computed independently with a privacy budget ϵ, then the data release consisting of the two queries, while

less private than any of the two, still satisfies meaningful guaranatees, (2ε)-DP in this example. This property allows to analyse the privacy of complex mechanisms by decomposing them into simple ones.

Proposition 2 (Post-processing [108]). *Let $\mathcal{M} : \mathcal{D} \to \mathcal{Y}$ be a randomised mechanism that is (ε, δ)-differentially private, and $f : \mathcal{Y} \to \mathcal{Y}'$ any randomised function. Then the mechanism $f \circ \mathcal{M}$ is also (ε, δ)-differentially private.*

In other words, post-processing the output of $\mathcal{M}$, e.g., via rounding, does not degrade its privacy guarantee. This is particularly important as it means that no adversary can "undo" the privacy guarantees.

Perturbing the true answer to a query $T(D, q)$ with carefully calibrated noise sampled from the Laplace distribution is enough to ensure ε-DP.

Definition 2.9 (Laplace distribution). *A random variable X taking values in $\mathbb{R}$ is said to follow the Laplace distribution with parameters μ and b $X \sim Lap(\mu, b)$, if its probability density function is equal to:*

$$\forall x \in \mathbb{R}, pdf(Lap(\mu, b))(x) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}}$$

The expected value of the random variable is equal to μ and its variance is equal to $2b^2$.

Theorem 2.1 (The Laplace mechanism is ε-DP [107]). *Denote by $R_{DPLaplace(\varepsilon)} : \mathcal{D} \times \mathcal{Q} \to \mathbb{R}$ the following randomised mechanism:*

$$R_{DPLaplace(\varepsilon)}(D, q) = T(D, q) + L, \text{ where } L \sim Lap(0, \Delta q/\epsilon)$$

Here, Δq is called the global sensitivity of the query and is defined as $\max_{D,D'} |T(D, q) - T(D', q)|$ computed over all the pairs of neighboring datasets. Then $R_{DPLaplace(\varepsilon)}$ satisfies ε-DP.

Intuitively, the scale of the noise that needs to be added to the answer to a query depends on the maximum impact of adding (or removing) any record. In this work, we will restrict to counting queries whose result can differ by at most 1 for any two neighboring datasets, i.e., $\Delta q = 1$. We refer the reader to Dwork et al. [107] for a proof of the theorem.

Finally, a popular alternative is the Gaussian distribution:

$$\forall x \in \mathbb{R}, pdf(N(\mu, \sigma^2))(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{\sigma^2}}$$

Setting, as before, the mean of the noise to $\mu = 0$ and carefully calibrating its variance σ^2, allows to satisfy (ε, δ)-DP. We refer the reader to Dwork and Roth [108] for the details.

Chapter 3

On the identifiability of interaction datasets across time

The work presented in this chapter is based on the following peer-reviewed journal article: Creţu, A.-M., Monti, F., Marrone, S., Dong, X., Bronstein, M., and de Montjoye, Y.-A. (2022). "Interaction data are identifiable even across long periods of time." Nature Communications, 13(1), 313.* `https://doi.org/10.1038/s41467-021-27714-6`

3.1 Introduction

An increasing fraction of our online and offline interactions is now captured by technology [191]. Large amounts of interaction data are now collected by messaging apps, mobile phone carriers, social media companies, and other apps to operate their service or for research purposes. Interaction data typically consist of the pseudonyms of the interaction parties, the timestamp of the interaction, and possibly further information. Mobile phone interaction data have been used to study the linguistic divide in a country [38], to study the interaction patterns of individuals with close connections over time [286], or to forecast the spatial spread of epidemics [34]. Similarly, interaction data have been used to study the spread of misinformation on Twitter [294, 223], the characteristics of news retweet networks during elections [44], or the effect of

Facebook friendship ties in political mobilisation [42]. Finally, close-proximity interaction data have been collected using Bluetooth to study human behaviour [111, 11, 285] and have recently been at the core of COVID-19 contact tracing apps aiming to help control the spread of the disease.

Despite previous claims [324, 4], interaction data are deeply personal and sensitive. They record with high precision who we talk to or meet, at what time, and for how long. Sensitive information can furthermore often be inferred from interaction data. Previous research, for instance, showed how algorithms can predict who a person's significant other is [15], their wealth [40, 207], demographics [59, 117], propensity to overspend [306], personality traits [91] and other attributes [133] from interaction data. Some works even leveraged homophily or network ties when making predictions [160]. Legal scholars and privacy advocates have long argued that interaction data are as sensitive as the content of the communication and that "metadata are data" [112, 289]. Mobile phone metadata have been at the core of the Snowden revelations and their collection was later deemed illegal in ACLU v. Clapper [142, 315]. More recently, the proportionality of contact tracing apps developed in the context of the COVID-19 pandemic has been questioned [157, 299, 239].

Interaction data can be shared or sold to third parties without users' consent, so long as they are anonymised. According to current data protection regulations such as the European Union's General Data Protection Regulation (GDPR) [2], or the California Consumer Privacy Act (CCPA), anonymised (or de-identified) data are no longer considered as personal data. The European Data Protection Board (EDPB) predecessor, the Article 29 Working Party, defined anonymisation as resistance to singling out, linkability, and inference attacks [19]. In particular, the linkability criterion refers to "the ability to link, at least, two records concerning the same data subject." While guidances are subject to the interpretation of the courts, matching identities between two pseudonymous datasets would likely mean that they are not anonymous under GDPR. Both legislations emphasise that personal data should not be stored for longer than necessary and then deleted or anonymised, with terms of service suggesting the latter to be common practice [48, 242, 114].

Re-identification attacks have long been used to identify individuals in pseudonymised datasets, where direct identifiers of a record such as personally identifiable information are replaced with random identifiers. In one seminal study, zip code, birth date and gender were used to identify the Governor of Massachusetts William Weld [320]; in another, the movies people had watched were used [227]. In 2013, it was shown that four points, approximate places and times, were enough to uniquely identify someone in location data 95% of the time [90], with formal similarity measures being proposed for approximate matching [276]. Numerous attacks have been proposed for de-anonymising social network data, both using exact [25, 148, 366, 370, 64, 345, 202, 132] or approximate [228, 238, 297, 168, 143, 296, 295, 311] matching information. Graph matching [259, 352, 184, 115] and anchor link prediction [210, 364] are two closely related problems. These attacks have called into question the anonymity of datasets and the ability of pseudonymisation to properly de-identify data.

A shortcoming of re-identification attacks is their reliance on auxiliary information that is fairly stable over time (gender, zip code, etc.) or from the same time period (spatio-temporal points, movies watched, etc.) as the one recorded in the dataset. For such *matching* attacks to be applicable to interaction datasets, an adversary would have to know the interactions of an individual in the same time period as the dataset. This assumption is restrictive and may be used to dismiss the risk posed by re-identification attacks in practice. However, an adversary may know how the individual behaves in general as determined in an observation period distinct from the one recorded in the dataset. For example, consider the case of an individual changing their direct identifier over time, e.g., mobile phone number, to hide their identity. The auxiliary information available to an adversary aiming to re-identify this individual is likely non-matching, and would consist of their interaction data in a different time period, when the individual used a different phone number.

Another shortcoming of matching attacks is that they are not robust to re-pseudonymisation strategies, where the pseudonyms replacing a direct identifier are renewed frequently over time, e.g., on a weekly basis. Re-pseudonymisation strategies aim to limit *by design* the amount of information recovered when a matching attack succeeds as well as the time period over which an adversary can collect the auxiliary information necessary to run the attack. However, the

possibility remains that re-pseudonymisation strategies may be thwarted by attacks aiming to piece together interaction data released under different pseudonyms of the same individual. Should such an attack succeed, a matching attack could again be used by an adversary to re-identify the individual.

Intuitively, we expect interaction data to have some degree of regularity as many of us communicate in a predictable fashion, e.g., we call one person every morning and another person every Saturday. In a small-scale study of mobile call data collected from 24 students, Saramaki et al. [286] indeed found the students' interaction patterns to be robust over an observation period of 18 months even in the presence of network turnover. However, it is unclear whether our interaction patterns are unique enough – in addition to stable – so as to enable large-scale identification across time.

Contribution. We here address these shortcomings by proposing a profiling attack for pseudomised and frequently re-pseudonymised interaction data. Unlike matching attacks, our profiling attack uses auxiliary information from one time period to profile and identify an individual in another non-overlapping time period. This makes it more broadly applicable, as the auxiliary data does not have to come from the same time period as the dataset. More specifically, we develop a profiling attack aiming to match an individual's k-hop interaction graph (i.e., the graph on interactions between the individual and its contacts, of contacts with their respective contacts, up to k hops) to a dataset of k-hop pseudonymous graphs from a different time period.

We use geometric deep learning [46] to learn behavioural profiles that are stable over time from people's interaction data. More specifically, we use a graph attention neural network [338] to build a vector representation (embedding) of an individual's *weekly* k-hop interaction graph. We optimise the embeddings for identification across time using the triplet loss [290]. Then, we use nearest neighbor matching on the embeddings to retrieve the individual in the dataset that most closely matches the target individual.

We evaluate our attack on a mobile phone dataset of more than 43.6K people. Using a 1-week gap between the dataset and the adversary's auxiliary information about the target individual, we find that our model is able to correctly identify a person 52% of the time based on their

2-hop interaction network ($k = 2$). Using only a person's interactions with their direct contacts ($k = 1$), our model can still identify them 15% of the time. We further show that the accuracy of our model only decreases slowly as time passes with 24% of the people still being correctly identified after 20 weeks ($k = 2$), thus making identification a real risk in practice.

We then show that our general graph profiling approach can be applied to other types of interaction data. We apply our model to Bluetooth close-proximity data similar to the one collected by COVID-19 contact tracing apps for more than 500 people and show that it is able to link together 1-hop interaction networks with 26% accuracy.

Our results provide evidence that disconnected and even re-pseudonymised interaction data remain identifiable even across long periods of time. [1] They also strongly suggest that current practices may not satisfy the anonymisation standard set forth by the EDPB in particular with regard to the linkability criteria.

3.2 Attack model

We consider a service S collecting data about the interactions it is mediating. We denote by $\mathcal{I}$ the set of individuals taking part in the communications recorded by S. For example, $\mathcal{I}$ could be the set of users of a contact tracing or messaging application or the subscribers of a mobile phone carrier and their contacts. We call interaction data the record describing the interaction between two individuals using S, consisting of the pseudonym of the two individuals, a timestamp, and sometimes other information. We define a time period $\mathcal{T} = [t, t')$ as the set of all timestamps between a start t (inclusive) and end t' (exclusive). Given a time period $\mathcal{T}$, we define the interaction graph $G_{\mathcal{T}}$ as the directed multigraph with node set $\mathcal{I}$ and an edge

[1] Our use of "long" for gaps of up to 20 weeks should be understood in the context of previous behavioral identification studies [227, 90, 92] which assumed no time gap between the adversary's auxiliary information and the dataset. The slow rate of decrease in the identification rate suggests that a substantial fraction of individuals are likely to remain identifiable after *even longer time periods*. The study of very long gaps, e.g., decades is an interesting open question, yet the extent to which piecing together an individual's interaction data across decades is a feasible experiment and would cause harm is unclear to us. Indeed, datasets have to be deleted or "anonymised", i.e., at least pseudonymised, after certain periods to comply with data retention policies. In the former case, very old datasets would not be available in the first place, while in the latter case we did not find credible attack scenarios.

between two nodes for each interaction between the corresponding individuals at a timestamp in the time period $\mathcal{T}$. Each edge is endowed with additional data m describing the interaction. For example, if S is a mobile operator, m would be the timestamp, the type of interaction (i.e., call or text), its direction (i.e., which party initiated it), and the duration for calls (see Fig. 3.1). If S is a close-proximity app, m would be the timestamp and the strength of the signal. We denote by k-hop neighbour of a node $v \in \mathcal{I}$ any node $w \in \mathcal{I}$ such that the shortest path between v and w in $G_{\mathcal{T}}$ is of length k.

Given a time period $\mathcal{T}$, $i \in \mathcal{I}$ an individual and $k = 1, 2, \ldots$, we define the k-hop Individual Interaction Graph (k-IIG) $G^k_{i,\mathcal{T}}$ as the subgraph induced in $G_{\mathcal{T}}$ by the set of nodes situated on paths of length at most k starting at node i, excluding interactions between the k-hop neighbours themselves. We denote by i the originating individual of k-IIG $G^k_{i,\mathcal{T}}$. Fig. 3.1 shows an example of a 2-IIG.

Our attack model assumes (see Fig. 3.1) that a malicious agent, *the adversary*, has access to (1) a dataset $\mathcal{D} = \{G^k_{i,[t_1,t'_1)} \ : \ i \in \mathcal{I}'\}$ consisting of the k-IIGs of people in $\mathcal{I}' \subset \mathcal{I}$ from time period $\mathcal{T}_1 = [t_1, t'_1)$, as well as to (2) auxiliary data $G^k_{i_0,[t_2,t'_2)}$ consisting in the k-IIG of a known target individual $i_0 \in \mathcal{I}'$, coming from a disjoint time period $\mathcal{T}_2 = [t_2, t'_2)$ (i.e., $t'_1 \leq t_2$ or $t'_2 \leq t_1$). We further assume that the adversary knows, for each k-IIG, which node is at the centre of the k-IIG (originating node), and that the k-IIGs are pseudonymised, meaning that the direct identifier(s) of a node (e.g., phone number), are replaced with a pseudonym such as a random hash. Additionally, we assume that every node has a different pseudonym in each graph it appears in.

The adversary's goal is to find the target i_0 in $\mathcal{D}$, i.e., find the $G^k_{i,[t_1,t'_1)} \in \mathcal{D}$ such that $i = i_0$. If successful, the adversary is said to have identified i_0 and is able to retrieve all their interactions from time period $[t_1, t'_1)$. We denote by time delay the quantity $D = t'_2 - t'_1$.

A motivating example is where an adversary aims to match an identified target individual, e.g., Jane Doe, to one of the pseudonyms in a de-identified dataset $\mathcal{D}$ using auxiliary information from a later time period. However, having an actual identity attached to the auxiliary information is not necessarily a requirement for the re-identification to be successful and for the dataset $\mathcal{D}$ to

be considered personal data under the EU GDPR. Indeed, the Article 29 Working Party (the legal predecessor of the European Data Protection Board) in its guidance on anonymisation defines three criteria to evaluate whether a dataset is anonymous. The linkability criteria refers to "the ability to link, at least, two records concerning the same data subject". While guidances are subject to the interpretation of the courts, matching identities between two pseudonymous datasets would likely mean that they are not anonymous under the EU GDPR.

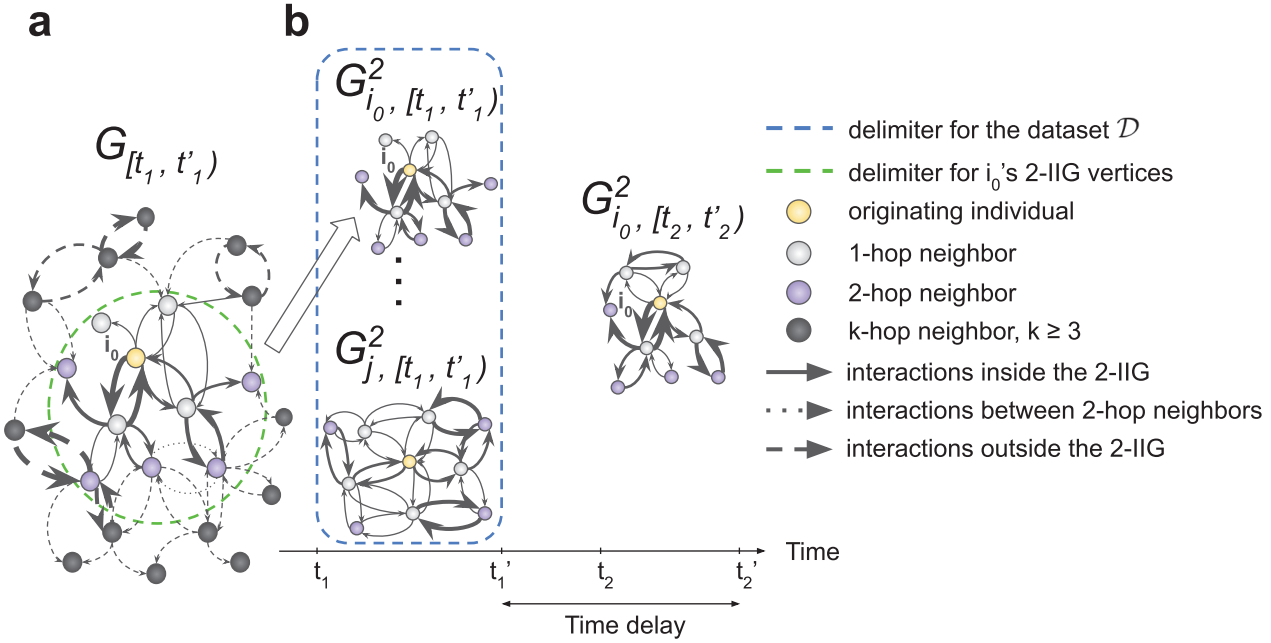
A more general motivation is to study the extent to which pseudonymised and frequently re-pseudonymised mobile phone and bluetooth interaction data are identifiable across time. Examples of attack scenarios include data collectors pseudonymising interaction data monthly as part of their data retention policy; poorly designed centralised contact tracing apps relying on frequent re-pseudonymisation to protect user's privacy; or the behavioural identification of a phone through e.g. its user's messaging pattern. The adversary could also be a law enforcement agency with, e.g., the Patriot Act giving intelligence agencies access to the 3-hop graphs of suspects (later restricted to 2-hop under the 2015 USA Freedom Act) [325].

Finally, we use a definition of the k-IIG that excludes interactions between the k-hop neighbours, as already done in the past for releasing mobile call graphs [39]. We consider this to be a realistic assumption, e.g., for $k = 1$ when the adversary's auxiliary information could come from the target's mobile. In the context of contact tracing, the adversary would have access to the log of the target's interactions with their contacts but would not have any information on the interactions between its contacts. This assumption makes results obtained using a method developed for k-IIGs a lower bound of what could be achieved with more information.

3.3 Attack methodology

3.3.1 Overview and justification of the method

Our goal is to develop a methodology for matching an individual's k-hop interaction graph to a dataset of k-hop graphs from a different time period that tightly estimates the risk of



c

Party A	Party B	Timestamp	Type	Direction	Duration (secs)
aSG64X	rxJKc9	2020-11-05 20:00:05	call	in	325
gvuQjU	dPefYb	2020-11-06 10:23:11	text	in	-
gvuQjU	dPefYb	2020-11-06 10:25:13	text	out	-
gvuQjU	LUrKAk	2020-11-06 10:47:20	call	out	124
⋮	⋮	⋮	⋮	⋮	⋮

Figure 3.1: **Setup of the behavioural profiling attack.** **a** An example of a 2-IIG highlighted in the larger graph it comes from. The vertices of the 2-IIG (inside the dashed green circle) are respectively the originating individual (in yellow), 1-hop neighbours (in gray), and the 2-hop neighbours (in purple). In solid lines are the edges that are part of the 2-IIG: all the edges between the originating and 1-hop neighbours; between the 1-hop neighbours; and between 1-hop and 2-hop neighbours, but excluding those between 2-hop neighbours (dotted lines). Dashed lines are all the other edges. For simplicity, all edges are shown as a single directed edge of thickness proportional to the total number of interactions. **b** The data available to the adversary consist of (left) 2-IIGs coming from time period $[t_1, t'_1)$, usually as part of an anonymised dataset, and (right) auxiliary 2-IIG data about a target individual A ($G^2_{i_0,[t_2,t'_2)}$). While we here display auxiliary data coming from a later period in time, our attack applies equally to cases where the auxiliary data comes from an earlier time period. **c** An example of mobile phone interaction data. Each interaction contains the pseudonyms of the parties A and B, timestamp, type of interactions, direction (equal to "out" if A initiated it, "in" otherwise) and the duration for calls. In this example, the person identified by "gvuQjU" received a text from another person, identified by "dPefYb", to whom the former responded 2 minutes later. After 22 minutes, "gvuQjU" called another individual, identified by "LUrKAk", for a duration of 124 seconds.

identification in practice. More specifically, our method should (1) frame the identification problem in a way that reflects practical risks, (2) scale to a large number of users in the dataset N, and (3) reflect or improve the state-of-the-art on k-hop graphs. We explain below the choices we make and the challenges that need to be addressed in order to achieve these goals.

Framing of the (re-)identification problem. We describe three main approaches for framing the (re-)identification problem, explaining their assumptions, analysing their strengths and weaknesses, and presenting justifications for the one we select.

The first framing of the re-identification problem, which is common in the social network de-anonymisation literature, is to find a mapping between the common users of two large graphs so as to maximise the number of correct matches [228, 238, 169, 168, 143, 296]. The goal is to study the risk of re-identifying users in a pseudonymised social network using another, partially overlapping, social network as the auxiliary information. In their seminal paper, Narayanan and Shmatikov [228] present a thorough motivation for this threat model, in particular stating that "auxiliary information which is likely to be available to the attacker is *global* in nature (*e.g.*, another social network with partially overlapping membership) and not restricted to the neighborhood of a single node". While we agree with this statement in the case of social networks, where large-scale auxiliary information may be collected by an adversary e.g., through scraping of Twitter, in the case of interaction data such as mobile phone data it is harder for adversaries to collect matching auxiliary information about *many users*. This concern is one of the motivations for our targeted threat model, which assumes that the auxiliary information consists of *one* target individual's k-hop interaction graph. The remaining question is whether methods developed for global de-anonymisation of two social networks can be applied to our problem. Two types of methods were developed for global de-anonymisation: seed-based [228, 311, 352, 238, 169, 168, 143] and seed-free methods [259, 296, 295]. The idea of seed-based methods is that the adversary knows, or reliably identifies prior to running the attack, a small number of nodes common between the two networks. Then, the adversary expands the scope of the attack iteratively by identifying new matches between the two networks based on the previously identified matches. Seed-based methods suffer from various limitations, such as the seed identification algorithm requiring specific structural assumptions to be met [228],

being sensitive to noise [238], and not scaling well [168] (see Sharad and Danezis [296] for a review), and are not directly applicable to our problem of piecing together fully non-matching k-hop graphs coming from different time periods. A few seed-free methods have also been proposed. In particular, Sharad and Danezis [296] train a binary classifier to predict if two nodes in different graphs belong to the same individual, then use it to identify the most likely matches as per the classifier's confidence. A set of initial, highly confident matches are refined and then used to select new matches using an iterative procedure. Sharad and Danezis [296] show their approach to outperform previous seed-based and seed-free methods. This approach was designed for social networks and it only exploits structural information about the graph. Our dynamic interaction graphs contain additional information, as there can be multiple edges for the same pair of nodes and every edge also contains timestamp information. We evaluate in Sec. 3.5.6 their binary classification method (introduced by the same authors in a previous paper [297]) and show it to perform worse than ours, even when using the additional information available in k-hop interaction graphs.

The second framing of the re-identification problem is targeted: the goal is to identify a user in a social network based on some matching auxiliary information. Backstrom et al. [25] proposed two active attacks, whereby an adversary injects a subgraph before the social network is published in pseudonymised form, by connecting fake nodes to a subset of targeted nodes. If the adversary can uniquely identify this subgraph, they can also re-identify the targeted nodes and learn which of these nodes are connected. Hay et al. [148] explored the use of structural information such as the degrees of a nodes' neighbors, and showed that some users can be uniquely identified based on this information. Similarly, Zhou et al. [366] modeled adversaries having knowledge of the target's neighborhood. Such adversaries are fundamentally limited, since their auxiliary information is assumed to match the one present in the dataset, an assumption that would not necessarily hold if the auxiliary information came from a different time period.

In cases such as ours where the auxiliary information is fully non-matching, the re-identification task can be naturally framed as an N-way classification task. The idea is to infer the correct individual to whom the auxiliary information belongs out of the N possible candidates by

training a classifier to perform the task [165, 183]. This framing is particularly common in authorship identification, where the goal is to identify the author of a text among a set of candidate authors based on their writing style [183, 226, 287, 249]. This involves creating a statistical model of the different candidate authors' writing patterns, usually by training a machine learning classifier on the authors' texts, and then applying it to new texts in order to predict their author. By design, this approach requires one or more samples from the candidate authors, which are different from the target sample whose authorship is unknown, in order to train the classifier. This requirement is satisfied in our case, making this approach appealing for tackling our problem. Indeed, the dataset contains k-hop samples from many individuals, including the target. These are fully non-matching with the auxiliary information that would be used at test time (since it comes from a disjoint time period) and can be used to train a classifier.

Finally, the third framing of the re-identification problem is as a binary classification task, where the goal is to infer, given one record whose identity is known and another record whose identity is disputed, whether they belong to the same individual (see, e.g., Sharad et al. [297] for the social network domain and Stamatatos for authorship verification in the text domain [313]).

Between the N-way classification framing and the binary classification framing, we select the former because it tells us how often an individual's k-hop graph can be matched among N candidates, naturally extending the success metric used in matching attacks, i.e., the fraction of users that can be uniquely identified [320, 91]. By contrast, the binary classification framing would only tell us how easy it is to distinguish between k-hop graphs that belong to the same individual and k-hop graphs that belong to different individuals. While this may be used to meaningfully quantify the risk in some settings, e.g., to verify whether a text belongs to a suspect, we find it less directly useful for identifying an individual among many candidates. Indeed, assume we use accuracy to quantify the success of the binary classifier, computing it on a balanced set of positive and negative pairs. How an accuracy of e.g., 80% translates to a practical risk is unclear to us. On the one hand, 80% is a good performance but on the other hand, it is an average success metric, meaning that the classifier may be good in distinguishing pairs of graphs of the same individual from pairs of graphs of different random individuals, but

may be less good on very similar records that belong to different individuals. In contrast, the accuracy of a method developed for the N-way classification approach directly translates to a real risk, i.e., if it succeeds 80% of the time against 10,000 individuals then 8,000 individuals can be identified.

To summarise, we here select the N-way classification framing, developing a *profiling attack* whose goal is to learn a distinguishing boundary between N individuals based on samples of their k-hop interaction graph. Our method uses state-of-the-art machine learning, as described later, and is explicitly designed for identifying k-hop graphs across different time periods. However, we acknowledge the value of simple baselines that do not rely on machine learning, computing simple features from the k-hop graphs and using them to identify the closest match of the target in the dataset. Such baselines are informative of the difficulty of the task: if even a simple baseline succeeds, e.g., 95% of the time against 10,000 individuals, then the task may not even require machine learning in the first place. In Sec. 3.5.6, we implement a range of simple baselines and show them to achieve poor performance, which strongly motivates the use of more advanced methods. In particular, we show our method to vastly outperform these approaches, highlighting the need to develop state-of-the-art attacks for tightly estimating privacy risks. Furthermore, for completeness and since our method relies on a notion of distance between the records, we also adapt it to and evaluate it in the binary classification setting (see Sec. 3.5.6 for details).

Scaling the method. We want our method to scale to a large number of individuals. If an attack only succeeds for small values of N, this means that it may only be considered a risk in the specific setting where the adversary can narrow down the candidates prior to running the attack. However, if an attack succeeds well even for very large values of N, e.g., in the tens of thousands or millions, this makes the attack a concern in practice. The scale that would make an attack a real risk has been debated [31]. Since large-scale sensitive datasets are usually not available to researchers, many identification studies only use small samples, e.g., less than a hundred or a few hundred identities to evaluate their attack methods [292, 333, 249]. This limitation has been used to dismiss the risk of re-identification attacks in practice. In one notable case, Barth-Jones et al. [31] characterised the re-identification study of de Montjoye et

al. [92] as unrealistic even if it worked 90% of the time on 1.1M credit card records, because it was not evaluated on a country scale. To bridge this gap, Farzanehfar et al. [116] have recently proposed a model to extrapolate the risk of re-identification from medium to country-scale location datasets.

It has been shown empirically that the success of re-identification attacks decreases with the number of individuals N in the dataset, see e.g., Tudor et al. for the smart meter domain [333], Kemelmacher at al. for the face domain [176], and Farzanehfar et al. for the spatio-temporal domain [116]. For matching attacks, Farzanehfar et al. [116] proved the general result that the expected fraction of unique individuals in a dataset of size N is a convex, non-increasing function of N. Their result captures the intuition that it becomes more likely for any two individuals in a dataset to share the same quasi-identifiers as the dataset size N increases. A similar argument applies to profiling attacks, which aim to learn a distinguishing boundary between N individuals. As N increases, the learning task becomes harder for the same number of training samples per class, because the training samples are less distinguishable for the same reason as above.

Our design of an N-way classifier is informed by the scaling requirement. In general, designing a machine learning classifier typically consists of two steps: (1) selecting a set of features to extract from the input and (2) selecting a loss function suitable for learning a distinguishing boundary between the N classes. We focus on step (2), which is essential for scaling the classifier to a large number of classes, abstracting for now step (1), i.e., the feature extraction process, as a function computing a vector representation (embedding) $\mathbf{h}(G^k_{i,\mathcal{T}}) \in \mathbb{R}^F$ from an individual i's k-hop interaction graph $G^k_{i,\mathcal{T}}$ in a time period $\mathcal{T}$. A popular loss function for classification problems is the cross-entropy loss. However, naïvely applying this loss function to our problem would result in a number of model parameters that scales linearly with the number of classes.[2]. For instance, to classify 10,000 individuals using $F = 32$ features, more than 320,000 parameters

[2]To see why, we describe briefly the typical implementation of the cross-entropy loss function. First, the embedding would be mapped to a vector of size N, e.g., through a linear layer $v = A\mathbf{h}(G^k_{i,\mathcal{T}}) + b \in \mathbb{R}^N$, with $A \in \mathbb{R}^{N\times F}$ and $b \in \mathbb{R}^N$ (although more complex functions such as a multilayer perceptron are also common). Second, the embedding would be converted to a probability vector via the softmax activation function: $y_l = \frac{e^{v_l}}{\sum_{l=1}^{N} e^{v_l}}$ (it is easy to verify that $\sum_{l=1}^{N} y_l = 1$). Third, the cross-entropy loss function would be computed as $-\log(y_i)$, where i is the individual from whom the input graph $G^k_{i,\mathcal{T}}$ is extracted, i.e., the correct class label.

would be needed for the output classification layer alone. Furthermore, each forward pass of the algorithm would also require $O(N)$ operations.[3] Apart from the computational cost, the signal provided by a handful of training samples for each class may not be sufficient to robustly train these parameters, which may lead to overfitting. To reduce the number of computations, hierarchical classification solutions have been proposed [224]. Another computationally efficient alternative is the triplet loss function [347]. We select this loss function because it was shown to be very successful in a task similar to ours, facial identification on a large scale [290, 153]. Note that we also experimented with the d-prime loss [319], which was originally designed for palmprint recognition, and found it to be less effective on our data.

The triplet loss function aims to optimise the parameters of an embedding function $\mathbf{h}$ so as to cluster together the embeddings of the same entity (e.g., an individual) computed on different instances or views of their data, while pushing them away from the clusters of different entities [347, 290, 28]. While in the facial identification task the different instances of the same individual are obtained using different images of their face [290], we here tailor the triplet loss to our task of identifying individuals across time based on their k-hop graphs. More specifically, we create different instances by extracting an individual's k-hop graph in different time periods. A triplet of k-IIGs $(G^k_{i,\mathcal{T}}, G^k_{i,\mathcal{T}'}, G^k_{i',\mathcal{T}''})$ contains data from two individuals $i \neq i'$, such that there are two k-IIGs from i, coming from time periods that are not equal, but could be overlapping $\mathcal{T} \neq \mathcal{T}'$, and a k-IIG from i' from a time period $\mathcal{T}''$ (not necessarily different from $\mathcal{T}$ or $\mathcal{T}'$). Let $\mathbf{h}(\mathbf{\Theta}) = \mathbf{h}(G^k_{i,\mathcal{T}}; \mathbf{\Theta})$, $\mathbf{h}^+(\mathbf{\Theta}) = \mathbf{h}(G^k_{i,\mathcal{T}'}; \mathbf{\Theta})$ and $\mathbf{h}^-(\mathbf{\Theta}) = \mathbf{h}(G^k_{i',\mathcal{T}''}; \mathbf{\Theta})$ denote the respective embeddings, where the embedding function is parameterised by the weights $\mathbf{\Theta}$. The triplet loss

$$\ell(\mathbf{\Theta}) = \max(0, \|\mathbf{h}(\mathbf{\Theta}) - \mathbf{h}^+(\mathbf{\Theta})\|_2 - \|\mathbf{h}(\mathbf{\Theta}) - \mathbf{h}^-(\mathbf{\Theta})\|_2 + \lambda) \tag{3.1}$$

tries to ensure that the profiles $(\mathbf{h}, \mathbf{h}^+)$ of the positive pair (i.e., the pair of profiles constructed from interaction data of the same individual, but different time periods) are closer than those $(\mathbf{h}, \mathbf{h}^-)$ of the negative pair (i.e., the pair of profiles constructed from i and another individual's interaction data in possibly, but not necessarily, different time periods $\mathcal{T}$ and $\mathcal{T}''$) by at least

[3] During training, this is needed in order to compute the denominator of the softmax activation function before backpropagation. At inference time, this is needed in order to determine the most likely class.

a margin λ. We average the triplet loss over a training set of positive and negative pairs and minimise it w.r.t the network parameters $\mathbf{\Theta}$. The optimal parameters $\mathbf{\Theta}^*$ obtained as the result of training are then used for the attack.

Once the embedding function is trained, we classify the target instance (here, the k-hop auxiliary graph) using nearest neighbor matching [347] to retrieve the individual in the dataset that most closely matches the target individual. More specifically, to identify the target individual $i_0 \in \mathcal{I}'$, our adversary computes the embedding of i_0 on its k-IIG in *target time period* $\mathcal{T}_2' \subset \mathcal{T}_2$, and the embeddings of all individuals in the dataset on k-IIGs in a *reference time period* $\mathcal{T}_1' \subset \mathcal{T}_1$ of same length as $\mathcal{T}_2'$. The adversary then computes the Euclidean distance $d_{i_0,j} = \|\mathbf{h}(G^k_{i_0,\mathcal{T}_2'}) - \mathbf{h}(G^k_{j,\mathcal{T}_1'})\|_2$ between the embedding of i_0 in target time period $\mathcal{T}_2'$ and the embeddings of all the individuals in the dataset $j \in \mathcal{I}'$ in reference time period $\mathcal{T}_1'$:

$$d_{i_0,j_1} \leq d_{i_0,j_1} \leq \ldots \leq d_{i_0,j_N},\ \{j_1,\ldots,j_N\} = \{1,\ldots,N\} \tag{3.2}$$

If the candidate with the smallest distance is (resp. R candidates with the smallest distance contains) the target individual (i.e., $i_0 \in \{j_1,\ldots,j_R\}$), we say that we have correctly identified i (resp. within rank R).

Throughout this work, we assume $\mathcal{T}_1$ (the dataset time period) and $\mathcal{T}_2$ (the auxiliary time period) to be at least one week long, although the approach can be generalised to any time lengths. Unless otherwise specified, we will use a reference and target time periods ($\mathcal{T}_1'$ and $\mathcal{T}_2'$) of one week, meaning that our attack exploits the regularity of weekly patterns in human behaviour.

State-of-the-art method for k-hop graphs. The remaining challenge in this work is constructing an embedding function $\mathbf{h}$ that effectively aggregates the information available in the k-IIG and allows the adversary to separate the profiles of different individuals. Recall that our k-IIGs are multigraphs where the nodes have no features and the edges are labelled with timestamp information, as well as other metadata, e.g., interaction type for mobile interaction data. A k-IIG contains topological information (the graph structure) as well as timestamp

and other information, both of which are likely to be relevant for the identification task. Unfortunately, the k-hop graph is not readily encoded as a list of numerical features that could be fed into a classifier or used to perform distance or similarity-based matching. To address this challenge, the standard approach in the re-identification literature is to manually design hand-engineered features that are deemed useful in fingerprinting individuals, i.e., that encode their unique patterns [183, 297, 226]. For instance, Sharad et al. extract the histogram of degrees of contacts of a node before which they feed into a random forest classifier [297]. In our experiments, we evaluate Sharad et al.'s method as well as other sets of hand-engineered features tailored to k-hop graphs, and find that they do not perform well (see Sec. 3.5.6 for the results). These results highlight the difficulty of manually extracting features from dynamic k-hop graphs optimal that perform well on the task of identifying people across time. It is indeed hard to determine *a priori* the exact features making an individual's interaction patterns unique and stable across time.

Recent developments in deep learning provide an alternative to hand-engineered feature extraction, in the form of *representation learning* [192]. The idea is to feed the *raw data* directly into a sufficiently expressive model, e.g., a deep neural network, and then optimise the parameters of the model for the task, e.g., classification [192]. Importantly, the feature extraction part is automated as the parameters of the model are optimised using gradient descent and back-propagation. Representation learning has achieved state-of-the-art performance on many image processing and natural language processing tasks, replacing hand-engineered feature-based approaches. An active area of research is the design of neural network architectures that improve performance on a given data domain and task. For instance, in the image domain, convolutional neural networks perform better than multilayer perceptrons [192, 105], while in the text domain, recurrent neural networks were widely used until the development of Transformers [336].

Historically, it has been challenging to use deep learning for graphs because, unlike images or acoustic signals, graphs have a non-Euclidean structure, i.e., informally speaking are not readily encoded as numbers. However, generalisations of deep learning architectures (in particular, convolutional neural networks) have recently been proposed for graph-structured data [47, 95, 181, 46], with successful applications to biology [146, 369, 340, 139, 136, 130], medicine [309],

and social network analysis [223, 146].

Inspired by these developments, we aim to construct a graph neural network embedding function. However, existing approaches are primarily designed for simple graphs whose nodes are endowed with features, while our k-IIGs are multigraphs whose nodes have no features and whose edges are endowed with timestamp and other information. To address this challenge, we develop a two-step approach.

First, we pre-process each k-IIG $G^k_{i,\mathcal{T}}$ to transform it into a simple graph $\bar{G}^k_{i,\mathcal{T}}$, referred to as the *simplified k-IIG*, where each node j is endowed with initial features $\mathbf{h}^{(0)}_j$. Our procedure, which we describe in Sec. 3.3.2, aggregates the 1-hop information available for each node, e.g., timestamp and contact graph, into a set of features [93]. We discard the nodes with incomplete information, which are the k-hop neighbors of i or non-subscribers to the service, which leaves us with a $(k-1)-$hop multigraph. We then convert this graph into a simple graph by discarding repeated edges between the same pair of nodes.

Second, since by design, the simplified k-IIG is a $(k-1)$-hop graph, when computing the embedding function we distinguish between the case $k = 1$ where the simplified k-IIG is a single node (endowed with features) and the case $k \geq 2$. More specifically, we use as the embedding function a multilayer perceptron (MLP) for $k = 1$ and a multi-layer graph neural network of the following form for $k \geq 2$:

$$\mathbf{h}^{(s)}_i = \xi^{(s)} \left(\left[\mathbf{h}^{(s-1)}_i, \sum_{j \in \mathcal{N}(i)} \alpha^{(s)}_j \mathbf{h}^{(s-1)}_j \right] \right) \tag{3.3}$$

$$\alpha^{(s)}_j = \frac{\alpha^{(s)}(\mathbf{h}^{(s-1)}_i, \mathbf{h}^{(s-1)}_j)}{\sum_{l \in \mathcal{N}(i)} \alpha^{(s)}(\mathbf{h}^{(s-1)}_i, \mathbf{h}^{(s-1)}_l)} \tag{3.4}$$

where the output $\mathbf{h}^{(s-1)}_i$ of layer $s-1$ is passed as the input to layer $s = 1, \ldots, S$. For each layer $1 \leq s \leq S$, $\xi^{(s)}$ is a non-linear parametric function implemented as an MLP with one hidden layer, followed by $\mathbb{L}_2$-normalisation. Finally, $\alpha^{(s)}$ denotes the attention weight computed as a nonlinear parametrised function of the features of node i and its neighbour $j \in \mathcal{N}(i)$ (see Sec. 3.3.3 for details). The neural attention mechanism, previously shown to

improve performance in tasks such as object recognition [222] and machine translation [27], has been adapted for graph inputs by aggregating a node's neighbourhood features via a weighted average over the features of the neighbours [338]. The attention weights are potentially different for distinct neighbours and are optimised for a specific learning task. We apply the embedding function to the input node-wise features $\mathbf{h}_i^{(0)}$ and its output $\mathbf{h}_i^{(S)} = \mathbf{h}(G_{i,\mathcal{T}}^k; \mathbf{\Theta})$ is used as the embedding of individual i, with $\mathbf{\Theta}$ denoting the network parameters of $\xi^{(s)}$ and $\alpha^{(s)}$ that are optimised during training. Note that while – to simplify the notation – we write $\mathbf{h}(G_{i,\mathcal{T}}^k; \mathbf{\Theta})$, the neural network is in fact applied to the simplified k-IIG, $\bar{G}_{i,\mathcal{T}}^k$.

We also experimented with the more general Message Passing Neural Network (MPNN) architecture [135], which we found to obtain similar performance (see Sec. 3.7 for a description and results). We select our method because we find it simpler and more interpretable.

In the remainder of this section, we provide the complete details of how we preprocess the k-IIG (Sec. 3.3.2), how we compute the embeddings (Sec. 3.3.3), and finally how we optimise them for the identification task (Sec. 3.3.4).

3.3.2 Preprocessing of a k-IIG

Let $G_{i,\mathcal{T}}^k$ denote the k-IIG of individual i in a week $\mathcal{T}$. Recalling that $\mathcal{I}$ denotes the set of individuals taking part in communications recorded by S, we denote by $\mathcal{I}'' \in \mathcal{I}$ the subset of subscribers to the service. We refer to other individuals in $\mathcal{I} \setminus \mathcal{I}''$ as non-subscribers, or out-of-network individuals. A particularity of mobile phone k-IIGs is that they record interactions between i, a subscriber to the service S and its contacts, which can be either subscribers or non-subscribers to S. Then, the k-IIG further records the interactions between subscriber contacts and their contacts, which as before, can be subscribers or non-subscribers to S. The same principle applies recursively up to $(k-1)$-hop contacts. Hence, a k-IIG contains all the interactions of subscriber nodes situated at most $k-1$ hops away from i. However, it only contains a subset of interactions of subscriber nodes situated k hops away from i, which are the interactions with subscribers situated at most $k-1$ hops away from i. Furthermore, a k-IIG contains interactions of a non-subscriber node only when the other party is a subscriber

node situated at most $k-1$ hops away from i. In particular, the k-IIG does not contain any interaction between non-subscriber nodes.

The distinction between subscriber and non-subscriber nodes is reflected in the data collection process, as each interaction is recorded as two edges when both parties are subscribers and as one edge when only one party is a subscriber. More specifically, an edge is described by the two parties v and $w \in \mathcal{I}$, and the interaction data $m = (t, d, t', \delta)$ consisting of the interaction's type t (call or text), the direction d (incoming or outgoing), the timestamp t' (in seconds), and the duration δ (in seconds if a call, or equal to $\perp$ otherwise). For an interaction between subscribers, i.e., $v, w \in \mathcal{I}''$, two directed edges are recorded $(v, w, m = (t, d, t', \delta))$ and $(w, v, m' = (t, d', t', \delta))$ with $\{d, d'\} = \{\text{in, out}\}$. The direction (either in or out) is determined by which party initiated the interaction: for example if w calls v, $d = \text{in}$ and $d' = \text{out}$. The other interaction data in m and m' (i.e., the interaction type t, the timestamp t', and the duration δ) are the same. For an interaction between a subscriber and a non-subscriber, e.g., $v \in \mathcal{I} \setminus \mathcal{I}''$ and $w \in \mathcal{I}''$, only one edge is recorded by S with the subscriber w as source node. For example, if v calls w for a duration of δ seconds, $(w, v, m = (\text{call}, \text{in}, t', \delta))$ is recorded but $(v, w, m = (\text{call}, \text{out}, t', \delta))$ is not recorded. A consequence is that non-subscriber nodes have only incoming edges, corresponding to the interactions with subscribers. Subscribers, on the other hand, have both incoming and outgoing edges. When examining a given k-IIG, we assume every node v having at least one outgoing edge (v, w, m) to be a subscriber.

To compute the embedding of a k-IIG, we first compute a set of features for the nodes using bandicoot [93], an open-source Python library developed for mobile phone metadata. Bandicoot has been used to predict people's personality [91], making it a suitable choice for identification. It takes as input an individual's list of interactions and outputs a range of features such as the number of voice and text contacts, call duration, etc. Importantly, *exact* features can only be computed for the subset of nodes in a k-IIG having all of their interactions contained in the k-IIG. As described above, these are the nodes that are both (1) at most $k-1$ hops away from the original individual i and (2) subscribers to the service. We infer the subscriber status of a node by checking if it has at least one outgoing edge. For each inferred subscriber j in a k-IIG that is at most $k-1$ hops away from i, we use bandicoot to compute an initial feature

vector $\mathbf{h}_j^{(0)}$. After computing features for these nodes, we discard the nodes having incomplete interactions, i.e., k-hop nodes and non-subscribers. Indeed, computing features for such nodes would greatly increase the computational cost and memory requirements (quadratically with the number of nodes since the forward pass involves a multiplication by the adjacency matrix of the graph) for likely very limited benefits (these nodes have little and incomplete data, resulting in noisy features).

We use the `bandicoot.individual.all` method with default parameters [4] to extract 33 features aggregating a user's activity over one week. The features range from simple statistics (such as the number of voice and text contacts) to more sophisticated statistics (such as the percentage of an individual's contacts that account for 80% of their interactions). To these features the adversary adds the percentage of out-of-network call, texts, call durations and contacts based on the information available on the edges.

Table 3.1 lists the $F = 23$ features we keep after removing the features having too many zeros or missing values in the mobile phone interaction dataset described in Sec. 3.4.1. While the bandicoot features are not part of our contribution, for clarity and reproducibility we give below the explicit formulas used to compute the features in Table 3.1. Consider $G_{i,\mathcal{T}}^k = (V, E)$ and a node $j \in V$ that is both a subscriber and at most $k-1$ hops away from i. Let $L_j = \{(v, w, m = (t, d, t', \delta)) \in E \ : \ v = j\}$ and $C_j = \{w \ : \ \exists (v, w, m) \in E \text{ such that } v = j\}$ be the set of interactions, respectively unique contacts of j. We will use $\mathbb{I}$ to denote the indicator function and drop the user subscript for the interaction and contact sets: $L := L_j$, $C := C_j$. We define the contact projection function $c : L \to C$ mapping an interaction (v, w, m) to the contact w. Similarly, we define the type, direction, timestamp and call duration projection functions, mapping an interaction $l_1 = (v, w, (t_1, d_1, t_1', \delta_1)) \in L$ to its type $t(l) = t_1$, direction $d(l) = d_1$, timestamp $t'(l) = t_1'$ and duration $\delta(l) = \delta_1$, respectively.

We further define outofnetwork : $C \to \{\text{True}, \text{False}\}$ mapping a contact w to True if and only if it is inferred as out-of-network, as explained previously. We denote by $L_{\text{call}} = \{l \in L \ : \ t(l) = \text{call}\}$ the set of calls and by $L_{\text{text}} = \{l \in L \ : \ t(l) = \text{text}\} = L \setminus L_{\text{call}}$ the set of texts. We denote

[4] `https://github.com/computationalprivacy/bandicoot/blob/master/bandicoot/utils.py`

by $n = |L|$ the number of interactions, by $n_{\text{call}} = |L_{\text{call}}|$ the number of calls and by $n_{\text{text}} = |L_{\text{text}}|$ the number of texts. Last, we denote by $C_{\text{call}} = \{w \in C \ : \ \exists l \in L_{\text{call}} \text{ such that } c(l) = w\}$ the set of call contacts. In what follows, we assume–without loss of generality–that $n_{\text{call}} \geq 1$. Indeed, if this is not the case, a feature involving a division by n_{call}, $|C_{\text{call}}|$, or the total call duration, will be imputed with 0.

1. The percentage of call contacts that account for 80% of the user's total call duration. We denote by $\delta_w = \sum_{l \in L_{\text{call}}} \delta(l)\mathbb{I}(c(l) = w)$ the total call duration between j and another contact $w \in C$. The contacts are ordered decreasingly by δ_w to obtain $\delta_{w_1} \geq \ldots \geq \delta_{w_{|C_{\text{call}}|}}$. Let d the minimum number of contacts such that: $\sum_{1 \leq p \leq d} \delta_{w_p} \geq \left\lceil 0.8 \cdot \left(\sum_{1 \leq p \leq |C_{\text{call}}|} \delta_{w_p} \right) \right\rceil$. The feature is then defined as $f_1 = d/|C_{\text{call}}|$.

2. The percentage of calls initiated by the user, defined as $f_2 = \left(\sum_{l \in L_{\text{call}}} \mathbb{I}(d(l) = \text{out}) \right)/n_{\text{call}}$.

3. The entropy of call contacts. For each contact $w \in C_{\text{call}}$, we denote by $p_w = \left(\sum_{l \in L_{\text{call}}} \mathbb{I}(c(l) = w) \right)/n_{\text{call}}$ the proportion of call interactions between the user and the contact w. The feature is defined as $f_3 = \sum_{w \in C_{\text{call}}} -p_w \ln p_w$.

4. The percentage of calls that occurred between 7PM and 7AM. Let night : $L \to \{\text{True}, \text{False}\}$ map an interaction l to True if l started between 7PM and 7AM and to False otherwise (given by the timestamp $t'(l)$). The feature is defined as $f_4 = \left(\sum_{l \in L_{\text{call}}} \mathbb{I}(\text{night}(l)) \right)/n_{\text{call}}$.

5. The percentage of call contacts that account for 80% of the user's calls. We denote by $n_w = \sum_{l \in L_{\text{call}}} \mathbb{I}(c(l) = w)$ the number of calls between j and another contact $w \in C_{\text{call}}$. The call contacts are ordered decreasingly by n_w to obtain $n_{w_1} \geq \ldots \geq n_{w_{|C_{\text{call}}|}}$. Let d the minimum number of call contacts such that: $\sum_{1 \leq p \leq d} \delta_{w_p} \geq \left\lceil 0.8 \cdot \left(\sum_{1 \leq p \leq |C_{\text{call}}|} \delta_{w_p} \right) \right\rceil$. The feature is then defined as $f_5 = d/|C_{\text{call}}|$.

6. The number of initiated calls, defined as $f_6 = \sum_{l \in L_{\text{call}}} \mathbb{I}(d(l) = \text{out})$.

7. The number of call contacts, defined as $f_7 = |C_{\text{call}}|$.

8. The number of text contacts, defined as $f_8 = |C_{\text{text}}|$.

Table 3.1: **The list of the 23 features used for the mobile phone interaction data**. We also list the corresponding methods to compute the features using the bandicoot toolbox. A dash symbol (-) indicates that the feature was not computed using bandicoot.

	Feature description	**Bandicoot method**
1	Percentage of call contacts that account for 80% of the total call duration	`percent_pareto_durations`
2	Percentage of initiated calls	`percent_initiated_interactions`
3	Entropy of call contacts	`entropy_of_contacts`
4	Percentage of calls that occurred between 7PM and 7AM	`percent_nocturnal`
5	Percentage of contacts that account for 80% of the user's calls	`percent_pareto_interactions`
6	Number of outgoing calls	`number_of_interactions`
7	Number of call contacts	`number_of_contacts`
8	Number of text contacts	`number_of_contacts`
9	Number of incoming calls	`number_of_interactions`
10	Number of incoming texts	`number_of_interactions`
11	Number of days, between 1 and 7, when the user was active	`active_days`
12-13	Mean/standard deviation of the time between two consecutive calls recorded for the user	`intervent_time`
14-15	Mean/standard deviation of the balance of the user's call contacts, where the balance is the ratio between the number of outgoing calls with a contact and the total number of calls	`balance_of_contacts`
16-17	Mean/standard deviation of the number of calls with each contact	`interactions_per_contacts`
18-19	Mean/standard deviation of call durations	`call_duration`
20	Percentage of out-of-network calls	-
21	Percentage of out-of-network contacts	-
22	Percentage of out-of-network call duration	-
23	Percentage of out-of-network texts	-

9. The number of incoming calls, defined as $f_9 = \sum_{l \in L_{\text{call}}} \mathbb{I}(d(l) = \text{in})$.

10. The number of incoming texts, defined as $f_{10} = \sum_{l \in L_{\text{text}}} \mathbb{I}(d(l) = \text{in})$.

11. The number of days in a week when the user is active. Let $\text{day} : L \to \{\text{Monday}, \text{Tuesday}, \text{Wednesday}, \text{Thursday}, \text{Friday}, \text{Saturday}, \text{Sunday}\}$ map an interaction l to the day of the week when it started (given by the timestamp $t'(l)$). The feature is defined as $f_{11} = |\{\text{day}(l) \ : \ l \in L\}|$.

12-13. The mean and standard deviation of times elapsed between two consecutive call interactions. The calls are ordered increasingly by the timestamp to obtain $l_1, \ldots, l_{n_{\text{call}}}$. If $n_{\text{call}} = 1$, the feature cannot be computed (will be imputed with 0 throughout the experiments). If $n_{\text{call}} \geq 2$, the times elapsed between two consecutive calls are computed as $it_p = l_{p+1} - l_p$ for $1 \leq p \leq n_{\text{call}} - 1$. The features f_{12} and f_{13} are defined as the mean and standard deviation of $\{it_p \ : \ 1 \leq p \leq n_{\text{call}} - 1\}$. In particular, $f_{12} = (l_{n_{\text{call}}} - l_1)/(n_{\text{call}} - 1)$.

14-15. The mean and standard deviation of the balance of the user's call contacts. We denote by $b_w = \big(\sum_{l \in L_{\text{call}}} \mathbb{I}(c(l) = w)\mathbb{I}(d(l) = \text{out})\big)/n_{\text{call}}$ the ratio between the number of outgoing calls between user j and contact $w \in C_{\text{call}}$ and the total number of interactions. The features f_{14} and f_{15} are defined as the mean and standard deviation of $\{b_w \ : \ w \in C_{\text{call}}\}$.

16-17. The mean and standard deviation of the number of call interactions with each contact. The features f_{16} and f_{17} are defined as the mean and standard deviation of $\{n_w \ : \ w \in C_{\text{call}}\}$.

18-19. The mean and standard deviation of the call durations $\{\delta_l \ : \ l \in L_{\text{call}}\}$.

20. The percentage of out of network call interactions, defined as: $f_{20} = \big(\sum_{l \in L_{\text{call}}} \mathbb{I}(\text{outofnetwork}(c(l))\big)/n_{\text{call}}$

21. The percentage of out of network contacts, defined as $f_{21} = \big(\sum_{w \in C} \mathbb{I}(\text{outofnetwork}(w)\big)/n$.

22. The percentage of out of network call duration, defined as $f_{22} = \big(\sum_{l \in L_{\text{call}}} \delta(l)\mathbb{I}(\text{outofnetwork}(c(l)))\big)/\big(\sum_{l \in L_{\text{call}}} \delta(l)\big)$.

23. The percentage of out of network texts, defined as $f_{23} = \big(\sum_{l \in L_{\text{text}}} \mathbb{I}(\text{outofnetwork}(c(l)))\big)/n_{\text{text}}$.

3.3.3 Embedding of a k-IIG

The adversary computes an embedding of the k-IIG on a simple graph derived from the k-IIG multigraph, which we call *simplified k-IIG*. This is a *simple directed graph* obtained by removing the featureless nodes from the k-IIG and collapsing all directed edges between two remaining nodes into a single directed edge of same direction. Formally, the simplified k-IIG is defined as $\bar{G}^k_{i,\mathcal{T}} = (\bar{V}, \bar{E})$, with:

$$V_1 = \{v \in V \ : \ v \text{ is on a path of length at most } k-1 \text{ from node } i\} \tag{3.5}$$

$$V_2 = \{v \in V \ : \ \exists w \in V \text{ with } (v, w, m) \in E\} \tag{3.6}$$

$$\bar{V} = V_1 \cap V_2 \tag{3.7}$$

$$\bar{E} = \{e = (v, w) \in \bar{V} \times \bar{V} \ : \ v \neq w \wedge \exists (v, w, m) \in E\} \tag{3.8}$$

Fig. 3.2 shows an example of a simplified 2-IIG. We briefly describe the simplified k-IIG $\bar{G}^k_{i,\mathcal{T}}$'s structure for each k:

- $k = 1$: $\bar{G}^k_{i,\mathcal{T}}$ consists of node i only;
- $k = 2$: $\bar{G}^k_{i,\mathcal{T}}$ consists of node i and all its 1-hop neighbours that are also in $\mathcal{I}''$. There is an edge between two nodes if there exists at least one edge between them in the 2-IIG;
- $k = 3$: $\bar{G}^k_{i,\mathcal{T}}$ consists of node i, all its 1-hop and 2-hop neighbours that are also in $\mathcal{I}''$. There is an edge between two nodes if there exists at least one edge between them in the 3-IIG; For computational reasons, we downsample the set of 2-hop neighbours so as the neighbourhood size does not exceed a given threshold τ. To this end, we select the τ nodes with largest number of contacts in the 3-IIG. We fix threshold $\tau = 200$. The contacts are selected according to outgoing edges, which correspond to both incoming and outgoing interactions.

To aggregate the features in $\bar{G}^k_{i,\mathcal{T}}$, we use a graph neural network with attention. Alg. 1 details the feedforward algorithm to compute the embedding of a k-IIG, while Fig. 3.3 illustrates it

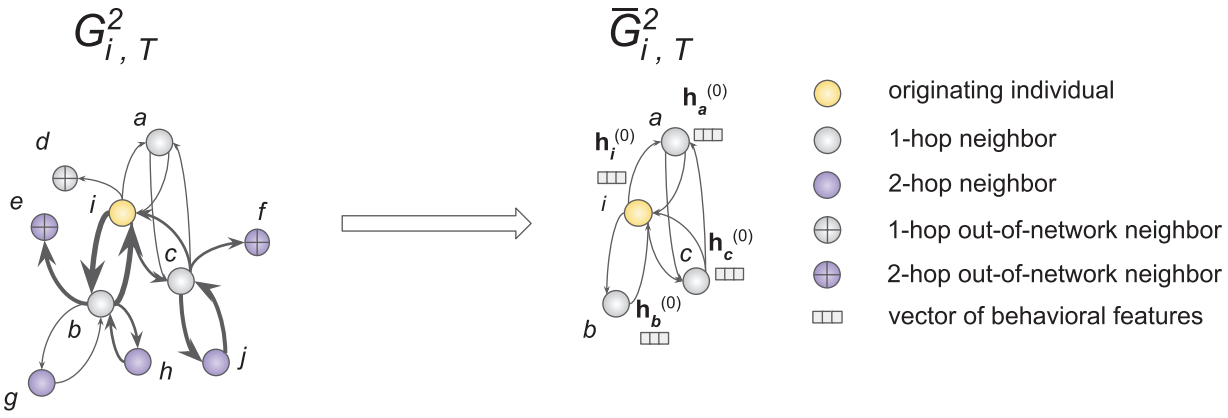


Figure 3.2: **An example of a simplified 2-IIG**. We show a 2-IIG, $G^2_{i,\mathcal{T}}$ (left), with vertex set consisting of originating individual i (yellow) and its 1-hop (a, b, c and d, gray) and 2-hop (e, f, g, h and j, purple) neighbours, and edges between them displayed as arrows with thickness proportional to the number of interactions. The nodes marked with $+$ (d, e and f) can be considered as out-of-network (i.e., $d, e, f \in \mathcal{I} \setminus \mathcal{I}''$), as they only have incoming edges in the 2-IIG. We further show the simplified 2-IIG $\bar{G}^2_{i,\mathcal{T}}$ (right), consisting of the originating individual i and the 1-hop neighbours, with one edge between any two nodes if there was at least one edge in 2-IIG. In the simplified 2-IIG, all nodes are endowed with behavioural features.

on an example of a 2-IIG. The algorithm takes as input the simplified k-IIG, $\bar{G}^k_{i,\mathcal{T}} = (\bar{V}, \bar{E})$, where each node $j \in \bar{V}$ is endowed with $\mathbf{h}_j^{(0)}$, the initial vector of behavioural features. For each node j in i's simplified k-IIG, we denote by $\mathcal{N}_{i,k}(j) = \{l \in \bar{V} \ : \ (j,l) \in \bar{E}\}$ their set of neighbours in the simplified k-IIG. Since the node representations are computed independently in each k-IIG, we drop the subscripts for convenience and use $\mathcal{N}(j)$ instead.

The network consists of a sequence of S layers, each performing a propagation step. This means that for every layer $s = 1, \ldots, S$, a new vector representation $\mathbf{h}_j^{(s)}$ is computed for each node j in the graph, by aggregating the features of the node's neighbours computed in the previous step, $\{\mathbf{h}_l^{(s-1)} \ : \ l \in \mathcal{N}(j)\}$. We use the attention mechanism of Veličković et al. [338] (lines 6-13). Attention computes a weighted average of the neighbour embeddings, where the weights can both (1) be learned and (2) depend on the node and neighbour. Then, we

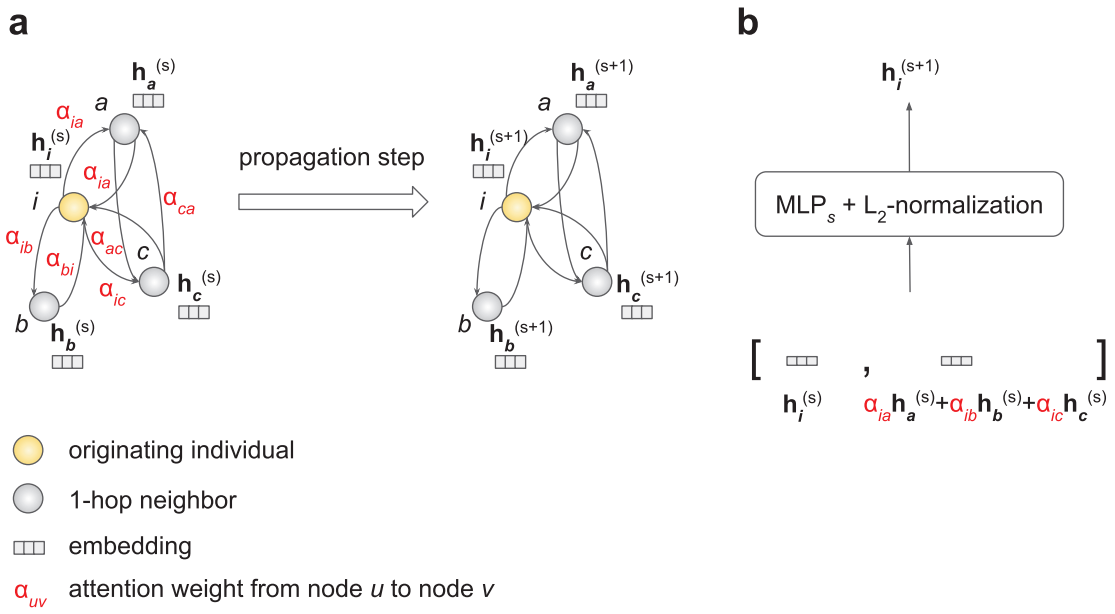


Figure 3.3: **One step of propagation to the neighbours on a simplified 2-IIG using graph attention.** **a** For each node in the 2-IIG, an embedding is computed at step $s+1$ using the embeddings at step s. The attention weights (denoted by α) are computed as shown in Alg. 1 (line 7), with superscript symbol s dropped here for illustration purposes. **b** The forward propagation step concatenates a node's embedding at step s to the weighted average of its neighbours' weights, to be given as input to an MLP layer followed by $\mathbb{L}_2$-normalisation.

concatenate the obtained representation to the node's features from the previous step $(s-1)$, and apply a non-linear transformation to the resulting vector. This propagation rule is similar to GraphSAGE [146], except that we use attention and a Multilayer Perceptron (MLP) as the transformation instead of a single fully connected layer (line 14), because it performs better on our problem. Finally, the output vector is $\mathbb{L}_2$-normalised, as we found it leads to better results.

3.3.4 Triplet sampling procedure

We optimise the embeddings for identification using the triplet loss [290]. To learn time-stable individual profiles that are well separated in the embedding space, we design a triplet sampling procedure. A triplet is composed of an anchor, a positive, and a negative example. The anchor and positive examples are two instances of the same individual, while the negative example is an instance of a different individual. For a given batch size B, our triplet sampling procedure:

Algorithm 1 Forward propagation algorithm for a k-IIG, $k \geq 2$

Inputs:
1. $\bar{G}^k_{i,\mathcal{T}} = (\bar{V}, \bar{E})$, the simplified k-IIG of an individual $i \in \mathcal{I}'$ from time period $\mathcal{T}$
2. Raw features $\mathbf{h}_j \in \mathbb{R}^F$ for each node $j \in \bar{V}$
3. $S \geq 1$, the number of layers. One layer does one step of propagation on the simplified 2-IIG to the neighbours.
4. Learnable Multilayer Perceptrons MLP_s with $2F_{s-1}$ input features and F_s output features, for $s = 1, \ldots, S$, with $F_0 = F$.
5. Learnable weight matrices $\mathbf{W}_1 \in \mathbb{R}^{F' \times F}$ and $\mathbf{W}_s \in \mathbb{R}^{F' \times F_{s-1}}$, for $s = 2, \ldots, S$.
6. Learnable vectors $\mathbf{v}_s \in \mathbb{R}^{2F'}$, for $s = 1, \ldots, S$.

Output:
$\mathbf{h}_i^{(S)}$, the feature vector of node i after the S propagation layers

Initialise:
$\mathbf{h}_j^{(0)} \leftarrow \mathbf{h}_j$, for every node $j \in \bar{V}$

for $s = 1$ to S **do**
  for $j \in \bar{V}$ **do**
    for l such that $(j, l) \in \bar{E}$ **do**
      `// Unnormalised attention weight for nodes j and l.`
      $a_{jl}^{(s)} \leftarrow \text{LeakyReLU}(\mathbf{v}_s^T [\mathbf{W}_s \mathbf{h}_j^{(s-1)}, \mathbf{W}_s \mathbf{h}_l^{(s-1)}])$
    end for
    `// Normalise the attention weights.`
    for l such that $(j, l) \in \bar{E}$ **do**
      $\alpha_{jl}^{(s)} \leftarrow \exp a_{jl}^{(s)} / (\sum_{(j,l) \in \bar{E}} \exp a_{jl}^{(s)})$
    end for
    $\hat{\mathbf{h}}_j^{(s)} \leftarrow \text{MLP}_s[\mathbf{h}_j^{(s-1)}, \sum_{(j,l) \in \bar{E}} \alpha_{jl}^{(s)} \mathbf{h}_l^{(s-1)})]$
    $\mathbf{h}_j^{(s)} \leftarrow \frac{\hat{\mathbf{h}}_j^{(s)}}{\|\hat{\mathbf{h}}_j^{(s)}\|_2}$
  end for
end for

1) samples one week i uniformly at random among the training weeks, 2) samples B individuals from $\mathcal{I}'$ and uses their k-IIGs in week i as anchor examples, and their k-IIGs in weeks $i-1$ and $i+1$ (modulo the number of training weeks) as positive examples and 3) for each anchor, selects a negative example via mini-batch moderate negative sampling [153]. For step 3, we consider as candidate negative examples all k-IIGs in weeks $i-1$, i and $i+1$ coming from the other $B-1$ individuals in the batch. In practice, each mini-batch contains $2B$ triplets, because at step 2) two different positive examples are considered for each anchor. We use mini-batch gradient descent for the optimisation. We define an epoch as a full pass over at least one anchor example of each individual in $\mathcal{I}'$.

3.4 Experimental setup

3.4.1 Mobile phone interaction dataset

We use a mobile phone interaction dataset composed of the 3-IIGs of $N = 43,606$ subscribers of a mobile carrier collected over a period of $T = 35$ consecutive weeks. The interactions contain the pseudonyms of the interacting parties, timestamp, the type of interaction (call or text), the direction (incoming or outgoing), and the duration for calls. All N subscribers interacted with at least 5 subscribers in each week. After discarding the outlier 3-IIGs of individuals having more than 350 subscriber contacts in all of the weeks, we are left with $N = 43,606$ 3-IIGs. The dataset contains a total of 10-100M unique identities and 100M-1B interactions.

We divide the time period into 35 consecutive weeks: $\mathcal{T} = \mathcal{W}_1 \cup \ldots \cup \mathcal{W}_T := \mathcal{W}_{1:T}$, where $\mathcal{W}_n = [t_n, t_{n+1})$ denotes the n-th week, with $1 \leq n \leq T$ and t_{n+1} and t_n differing by one week. Unless otherwise specified, in our experiments we use the following settings:

- The pseudonymised dataset $\mathcal{D}$ consists of the k-IIGs of all the N people from the first $T' = 15$ weeks of data, i.e., $\mathcal{T}_1 = \mathcal{W}_{1:T'}$. We set aside the last week in the dataset as the reference time period (reference week) $\mathcal{T}_1' = \mathcal{W}_{T'} = \mathcal{W}_{15}$ to which the embedding of the target individual is compared against (see Eq. 3.3.1). We train and validate the embedding model on data from the first 14 weeks.

- The auxiliary information available to the adversary on the target individual is the k-IIG from one of the remaining weeks $\mathcal{T}_2 \in \{\mathcal{W}_{T'+1}, \ldots, \mathcal{W}_T\}$. The attack is agnostic to whether this auxiliary information is identified or re-pseudonymised with respect to the dataset.

- The target time period (target week), which is used to perform Euclidean distance comparison against embeddings of individuals extracted from the reference week (see Eq. 3.3.1), is $\mathcal{T}_2' = \mathcal{T}_2$.

The metric of attack success is the probability of identification within rank R from the target

to the reference week. This is defined as the fraction of people among the N subscribers who are correctly identified within rank R.

Feature pre-processing. We provide more details on how we dropped features from the ones we initially extract using bandicoot (see Sec. 3.3.2). We computed the features of all N individuals on the first 14 weeks of data. We removed the features with at least 30% of missing values, with the exception of the number of incoming texts and the number of text contacts, which can be meaningfully imputed with zeros when the value is missing. We also removed the number of outgoing texts because it was equal to zero more than 99% of the time when the value was not missing.

Feature transformation. While some of the features are approximately normal, others are heavily skewed toward zero values, such as the number of outgoing calls. This is a known characteristic of mobile phone interaction data [293]. We use a Box-Cox transformation to convert the heavily skewed variables into an approximately normal distribution. More specifically, we transform the features 6-10 and 16-19 in Table 3.1 having skewness coefficient ≥ 3 (computed from the N originating individuals' features from the first 14 weeks of data after filling null values with zeroes) after adding 1 to ensure they are strictly positive. We then standardise all features to have zero mean and a variance of 1. To avoid information leakage from validation or test data to the trained model, we always fit transformations on nodes of the training dataset.

Other details. While the simplified k-IIG should always be symmetric (its nodes are all subscribers that, by definition, have both incoming and outgoing edges), sometimes it is not, due to artefacts in the data collection. This happens, for instance, when only one of the two edges corresponding to the same interaction is recorded, and no other interactions exist between these parties. We restore the symmetry by adding edges when they are missing.

3.4.2 Training setup

To obtain an unbiased estimate of the model performance, we evaluate the probability of identification on reference and target weeks *unseen during training*. Since we use the last week in the dataset as the reference week, this leaves us with $T' - 1 = 14$ weeks of data to train (and validate) the model.

Model parameters. To embed k-hop networks with $k \in \{2, 3\}$, we use an architecture with $S = 2$ layers, where MLP_1 has an input size of 46 (i.e., $2F_0$, with $F_0 = F = 23$), one hidden layer of size 128 and an output size of $F_1 = 50$ and MLP_2 has an input size of 100 (i.e., $2F_1$), one hidden layer of size 128 and an output size of $F_2 = 50$. For the parameters $\mathbf{v}_s$ and $\mathbf{W}_s$ ($s \in \{1, 2\}$) used to compute the attention weights (line 8 of Alg. 1), we use $F' = 20$. To embed 1-hop networks ($k = 1$), we apply an MLP directly to the feature vector $\mathbf{h}_i^0$ with an input size $F = 23$, a hidden layer of size 128 and output size of 50, followed by $\mathbb{L}_2$-normalisation. The hidden layer of MLPs is followed by the REctified Linear Unit (ReLU) nonlinearity. We implement the models in Pytorch 1.3.1 using Python 3.7.3.

Training procedure. We parametrise the triplet loss using a margin value of $\lambda = 0.25$. We train the models for a maximum number of 100 epochs, decreasing the learning rate η by a factor of 2 after 5 epochs of non-increasing probability of identification computed over the validation weeks. We stop training if the learning rate decreases below 10^{-5}. We use stochastic gradient descent optimisation with a mini-batch size of 64 and a weight decay penalisation of 0.001. We perform hyperparameter selection for the learning rate using cross validation, with possible values $\eta \in \{0.05, 0.025, 0.01\}$ for $k = 1$ and $\eta \in \{0.01, 0.005, 0.001\}$ for $k = 2, 3$.

Hyperparameter selection. For each $k \in \{1, 2, 3\}$, the adversary selects the best hyperparameters using cross validation on the weeks $\mathcal{W}_{1:T'-1}$, where each test fold is composed of two consecutive weeks (the reference and target week, respectively). With $T' = 15$ being odd, the $(T' - 1)/2 = 7$ disjoint test folds are defined as $\{(\mathcal{W}_{2i+1}, \mathcal{W}_{2i+2}), 0 \leq i < (T' - 1)/2\}$. In each fold, the previous two time periods (modulo $T' - 1$) are used as validation weeks for early stopping (with the probability of identification as the criterion), and the remaining weeks are

used for training.

Final model. Given the best hyperparameter set, the final model is trained on data from 12 consecutive weeks $\mathcal{W}_{1:T'-3}$, using validation weeks $\mathcal{W}_{T'-2}$ and $\mathcal{W}_{T'-1}$ for early stopping.

3.5 Results

3.5.1 Probability of identification after one week

We first evaluate our model with a time delay of one week between the reference and target time periods. We report the probability of identification within rank R, defined as the fraction of people among the N subscribers who are correctly identified within rank R (averaged over 10 runs).

Fig. 3.4 shows that our model correctly identifies people $p_{k=2} = 52.4\%$ of the time in a dataset of 43.6K people with $k = 2$ i.e., when the adversary has access to an individual's interactions as well as the interactions of their contacts. It also shows the probability of identification of a target individual within the top R matches. Our model is able to rank the correct person among the top 10 candidates $p_{k=2} = 77.2\%$ of the time and among the top 100 candidates $p_{k=2} = 92.4\%$ of the time.

When $k = 1$, i.e., when the adversary has only access to the individual's direct interactions, our model is still able to identify people $p_{k=1} = 14.7\%$ of the time. While having access to the 2-hop information helps, our model still performs much better than random for $k = 1$. The probability of identifying the correct person among the top 10 candidates (rank 10) is $p_{k=1} = 34.7\%$ while the rank 100 probability is $p_{k=1} = 61.9\%$ respectively. Interestingly, having access to information beyond the target's direct contacts ($k = 3$) only marginally increases the probability of correct identification $p_{k=3} = 56.7\%$ (a 7.9% increase w.r.t. $k = 2$). Higher ranks probabilities similarly increase to $p_{k=3} = 81.7\%$ and $p_{k=3} = 94.6\%$, respectively a 5.8% and a 2.4% increase. On the one hand, this marginal increase could be due to the fairly large number of nodes reached with $k = 3$ (121.5 ± 48.8 for $k = 3$ vs 17.3 ± 13.4 for $k = 2$) thereby limiting

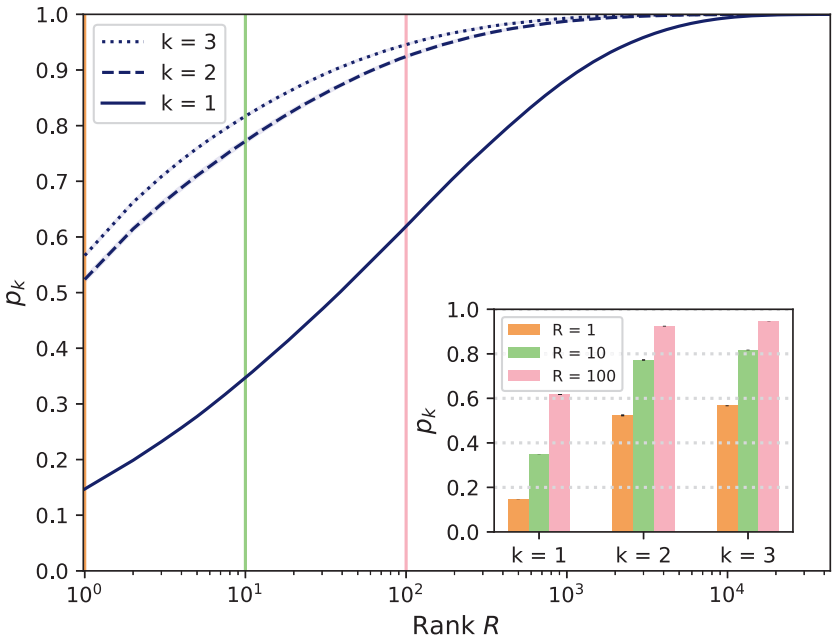


Figure 3.4: **Probability of identification after one week.** For each $k \in \{1, 2, 3\}$, we plot p_k, the probability of identification within rank $R \in \{1, \ldots, 43,606\}$ when the time delay is $D = 1$ week, with the 95% confidence interval shown in light blue. **(Inset)** shows the probability of identification for ranks 1, 10 and 100, with error bars for the 95% confidence interval.

the usefulness of data from larger k. On the other hand, this could also be due to our particular choice of architecture. In particular, while we downsampled the simplified k-IIG to contain no more than $\tau = 200$ nodes for $k = 3$, the graph neural network architecture might still suffer from oversmoothing. Given that new architectures could be developed to leverage information coming from the 3-IIG specifically, from a privacy perspective, our results are thus only a lower bound on the risk of re-identification.

3.5.2 Impact of time delay of auxiliary data and the dataset

The accuracy of our model is likely to decrease as time passes: people change behaviour, make new friends and lose contact with others. Fig. 3.5 shows that, despite this, the probability of correct identification only slowly decreases with the time delay $D = t'_2 - t'_1$ (see Sec. 3.2). Even after 20 weeks, our model still correctly identifies people $p_{k=2} = 24.3\%$ of the time when $k = 2$. This suggests that the profiles our model extracts from the data capture key behavioural

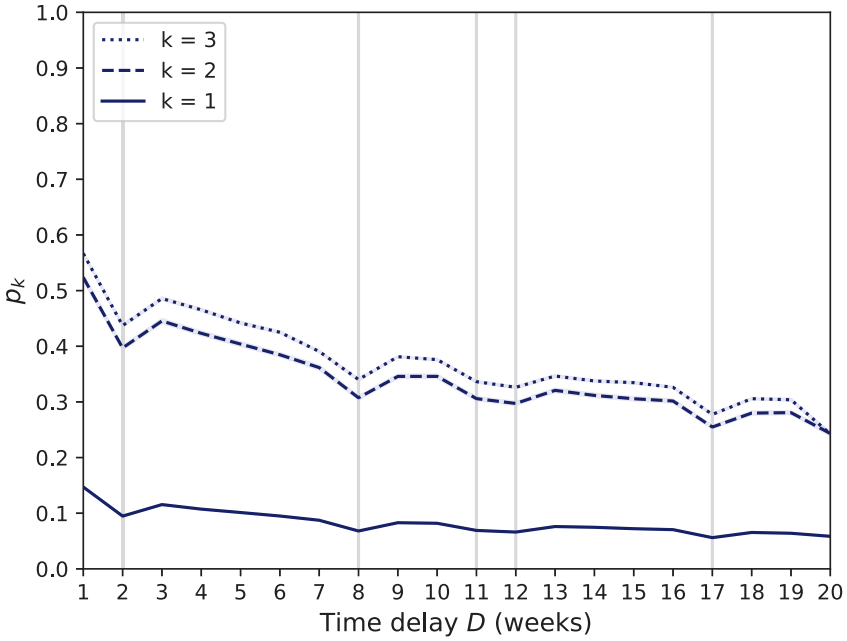


Figure 3.5: **Probability of identification when the time delay increases.** We plot p_k, the probability of identification within rank 1 for $k \in \{1, 2, 3\}$ when the time delay between the dataset and the adversary's auxiliary information is equal to D weeks. The auxiliary information is one week long. The 95% confidence interval is shown in light blue. The vertical grey lines correspond to holidays. The probability of identification decreases as $y = p_k(D = 1) - \alpha_k \times (D - 1)$, with $\alpha_1 = -0.006$, $\alpha_2 = -0.017$ and $\alpha_3 = -0.018$.

features of individuals. The probability of identification decreases similarly slowly with time for $k = 3$ and $k = 1$.

Interestingly, Fig. 3.5 shows that the probability of identification (p_k) visibly decreases when the time delay is of 8, 11, 12, and 17 weeks, respectively. In a post-hoc analysis, we found that they all correspond to weeks containing a national holiday. This further suggests that our model captures a person's routine weekly behaviour, both weekdays and weekend, and consequently loses some accuracy when a user's behaviour changes in response to external events.

3.5.3 Impact of time period length of auxiliary data

We have so far assumed that the adversary has access to a week of a target individual's data, i.e., their auxiliary information is the target individual's k-IIG from one week. In practice, an adversary might often have access to more weeks of data from an individual. In the D4D

challenge, data were for instance re-pseudonymised every two weeks [94] while a company wanting to archive transactional data might decide to pseudonymise and archive it on a monthly basis.

To simply evaluate the extent to which more auxiliary data increase accuracy, we combine the predictions from sequences of target weeks of growing length L, i.e., consider $\mathcal{T}_2 = \cup_{l=1}^{L} \mathcal{W}_{T'+l}$, for $L = 1, \ldots, T'-T$. For each length L, we combine the predictions from the $T'+1, \ldots, (T'+L)$-th weeks available as auxiliary data using a majority vote: the candidate that was ranked first most of the times is the final prediction. The tie-breaks are decided by the lowest total distance between the target individual and the highest ranked candidate.

Fig. 3.6 shows how having auxiliary data over several weeks further improves the performance of the attack. For $k = 2$, the probability of correct identification increases from $p_{k=2} = 52.4\%$ with one week of auxiliary data to $p_{k=2} = 66.0\%$ with $L = 16$ weeks. Interestingly, the probability of correct identification for all values of k increases fast and then plateaus around $L = 8$, even slightly decreasing after $L = 16$ and $L = 15$ for $k = 2$ and $k = 3$, respectively. Despite having access to more data, the attack is less accurate for increasing time delay. While this might seem surprising at first, we hypothesise that this is due to small changes to people's behaviour over time. This makes auxiliary data that are more distant in time less useful than closer one and sometimes slightly detrimental. The maximum probability for $k = 2$ is at $L = 16$ weeks ($p_{k=2} = 66.0\%$) and for $k = 1$ and $k = 3$ at $L = 20$ ($p_{k=1} = 19.4\%$) and $L = 13$ ($p_{k=3} = 69.3\%$) respectively.

3.5.4 Impact of the population size

To better understand how well the attack scales with the population size, we evaluate its performance when identifying $N' \leq N (N = 43,606)$ people among the same N' people. For this analysis, we use the final model trained on data from the N people. We sample ten random subsets of size N' uniformly without replacement and compute the average probability of identification $p_k(N')$ (defined as the fraction of people among the N' that are correctly identified within rank 1). We compute the probability of identification $p_k(N')$ for $N' = 2$, then

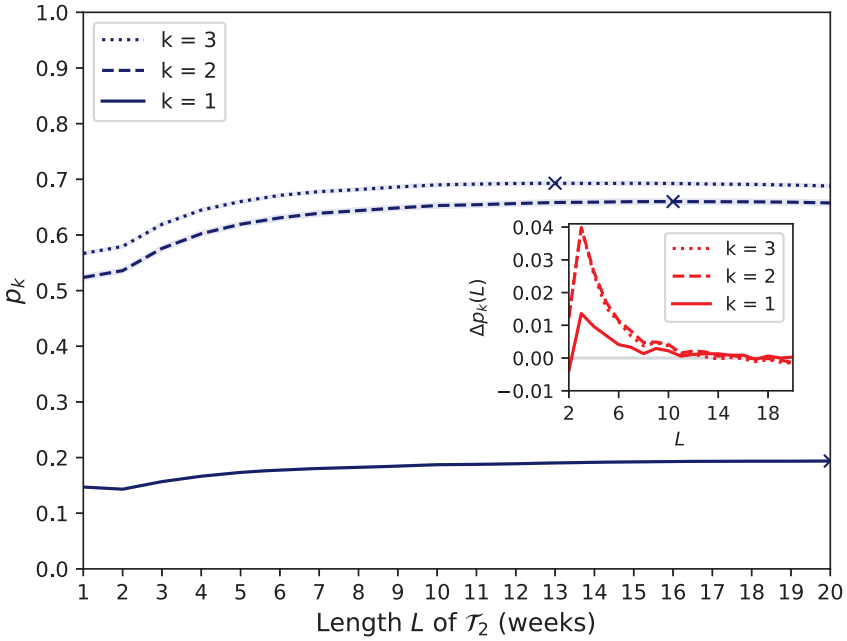


Figure 3.6: **Probability of identification for increasing time period length of auxiliary data.** For each $k \in \{1, 2, 3\}$, we plot p_k, the probability of correct identification ($R = 1$) when the adversary's auxiliary data $\mathcal{T}_2$ consist of L weeks, $1 \leq L \leq 20$ (the largest value for each k is marked). The 95% confidence interval is shown in light blue. **(Inset)** shows the difference quotient $\Delta p_k(L) = p_k(L) - p_k(L-1)$ for $2 \leq L \leq 20$.

for $N' = 1,000$ to $43,000$ using increments of $\Delta N = 1,000$, and finally for $N' = N$. The difference quotient is defined as $\Delta p_k(N') = (p_k(N') - p_k(N' - \Delta N'))/\Delta N'$.

Fig. 3.7 shows that the probability of identification decreases slowly with the reference population size, but that $\Delta p_k(N')$ decreases fast as N' grows. The difference quotient seems to be still decreasing around $N' = N$, suggesting that the probability of identification would decrease at an even slower rate for larger values of N'.

3.5.5 Post-hoc analysis of features impacting identification

We finally perform a post-hoc analysis to better understand who are the people that our model identifies correctly. To this end, we compute for each person the fraction of target weeks such that they are correctly identified by our attack using that week as auxiliary information. Fig. 3.8 shows (in blue) that for $k = 2$, 86.8% of people are correctly identified by our model at least

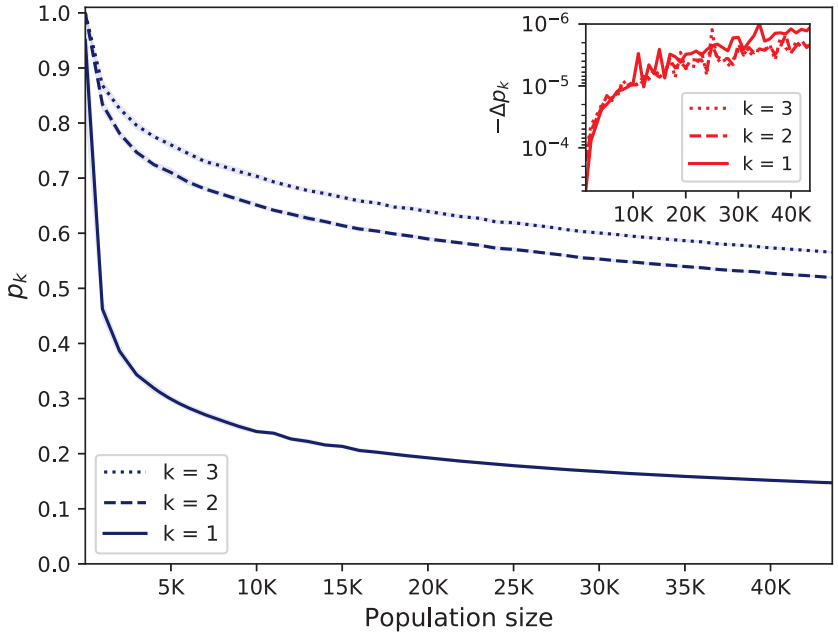


Figure 3.7: **Attack's performance with increasing population size.** We show $p_k(N')$, the probability of identification within rank 1 for $k \in \{1, 2, 3\}$ in a population of size N'. The 95% confidence interval is shown in light blue. **(Inset)** shows the negative difference quotient $-\Delta p_k(N') = -(p_k(N') - p_k(N' - \Delta(N))/\Delta N'$. The probability of identification decreases with the population size N', but at increasingly slower rates.

once (5% of the 20 target weeks).

We compare this with a naïve model in which individuals are identified independently in each week with the same probability as our attack, and independently from one another. In this setting, the number of weeks when a person is correctly identified follows a Poisson binomial distribution defined as the probability distribution of $B := \sum_{l=T'+1}^{T} B_l$ with $B_l \sim \text{Bernoulli}(p_l)$, where p_l denotes the probability of identification in target week l using our attack. We generate N i.i.d. samples of B, then compute the fraction $B/(T - T')$ of target weeks such that a person is correctly identified using that week as auxiliary information, under the naïve model.

Fig. 3.8 shows (in gray) the results of the naïve model averaged over 100 repetitions. We can see that our attack identifies some people in many more weeks than expected. For $k = 2$, the people we identify more often than expected are correctly identified in at least 40% of the weeks. The two curves cross one another at 20% and 45% for $k = 1$ and $k = 3$ respectively. For every $k \in \{1, 2, 3\}$, the lowest abscissa value where our approach outperforms the baseline

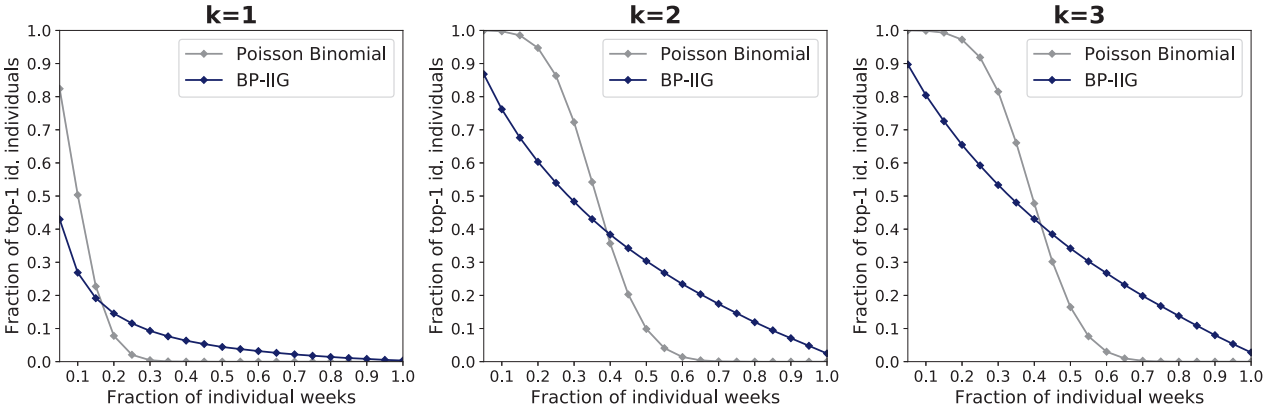


Figure 3.8: **Fraction of identified people vs fraction of individual weeks.** For each $k \in \{1, 2, 3\}$, we plot the fraction of people that are identified in at least a given fraction of individual weeks, using our model (in blue) and according to a Poisson binomial distribution (in gray, averaged over 100 trials). Our attack identifies 38.4% (resp. 14.5% and 38.5%) of the people more often than expected for $k = 2$ (resp. for $k = 1$ and $k = 3$).

is the same across attack initialisations.

We then divide the $N = 43,606$ people in the dataset into two groups based on their identifiability using these results. For each of the 10 runs of our attack ($k = 2$), we choose as cut-off the percentage f of individual weeks where our attack first outperforms the random baseline, thus assigning people to one of two groups: people who are identifiable in at least a fraction f of the weeks, and people who are identifiable in strictly less than a fraction f of the weeks. For all the 10 runs of our attack, the cut-off percentage where our model improves on the baseline is $f = 40\%$. For each individual, we use a majority vote on the labels assigned for the various runs in order to assign a single label.

Using a logistic regression, we examine what distinguishes the people our model identifies more often and less often than expected for $k = 2$. We train a model to predict the correct label using the features listed in Table 3.2, averaged over the entire set of $T = 35$ weeks. Differently from our attack methodology, we aggregate information from both calls and texts, unless a feature relates to calls specifically (e.g., the call duration). We did this because the text data is generally sparser than the call data and we wanted to focus on the general behavioural patterns. Similar formulas to those provided in Sec. 3.3.2 are used by bandicoot to compute the features.

Fig. 3.9 suggests that, when holding all other features constant, individuals with more inter-

Table 3.2: **The list of 19 features used in the post-hoc analysis.** We also list the corresponding methods to compute the features using the bandicoot toolbox. A dash symbol (-) indicates that the feature was not computed using bandicoot.

	Feature description	**Bandicoot method**
1	Percentage of call contacts that account for 80% of the total call duration	`percent_pareto_durations`
2	Percentage of initiated calls	`percent_pareto_interactions`
3	Entropy of contacts	`entropy_of_contacts`
4	Percentage of interactions that occurred between 7PM and 7AM	`percent_nocturnal`
5	Percentage of contacts that account for 80% of the user's interactions	`percent_pareto_interactions`
6	Number of outgoing calls	`number_of_interactions`
7	Number of contacts (call and text)	`number_of_contacts`
8	Number of days, between 1 and 7, when the user was active	`active_days`
9-10	Mean/standard deviation of the time between two consecutive interactions recorded for the user	`interevent_time`
11-12	Mean/standard deviation of the balance of the user's contacts, where the balance is the ratio between the number of outgoing interactions with a contact and the total number of interactions	`balance_of_contacts`
13-14	Mean/standard deviation of the number of interactions with each contact	`interactions_per_contacts`
15-16	Mean/standard deviation of call durations	`call_duration`
17	Percentage of out-of-network interactions	-
18	Percentage of out-of-network call durations	-
19	Percentage of out-of-network contacts	-

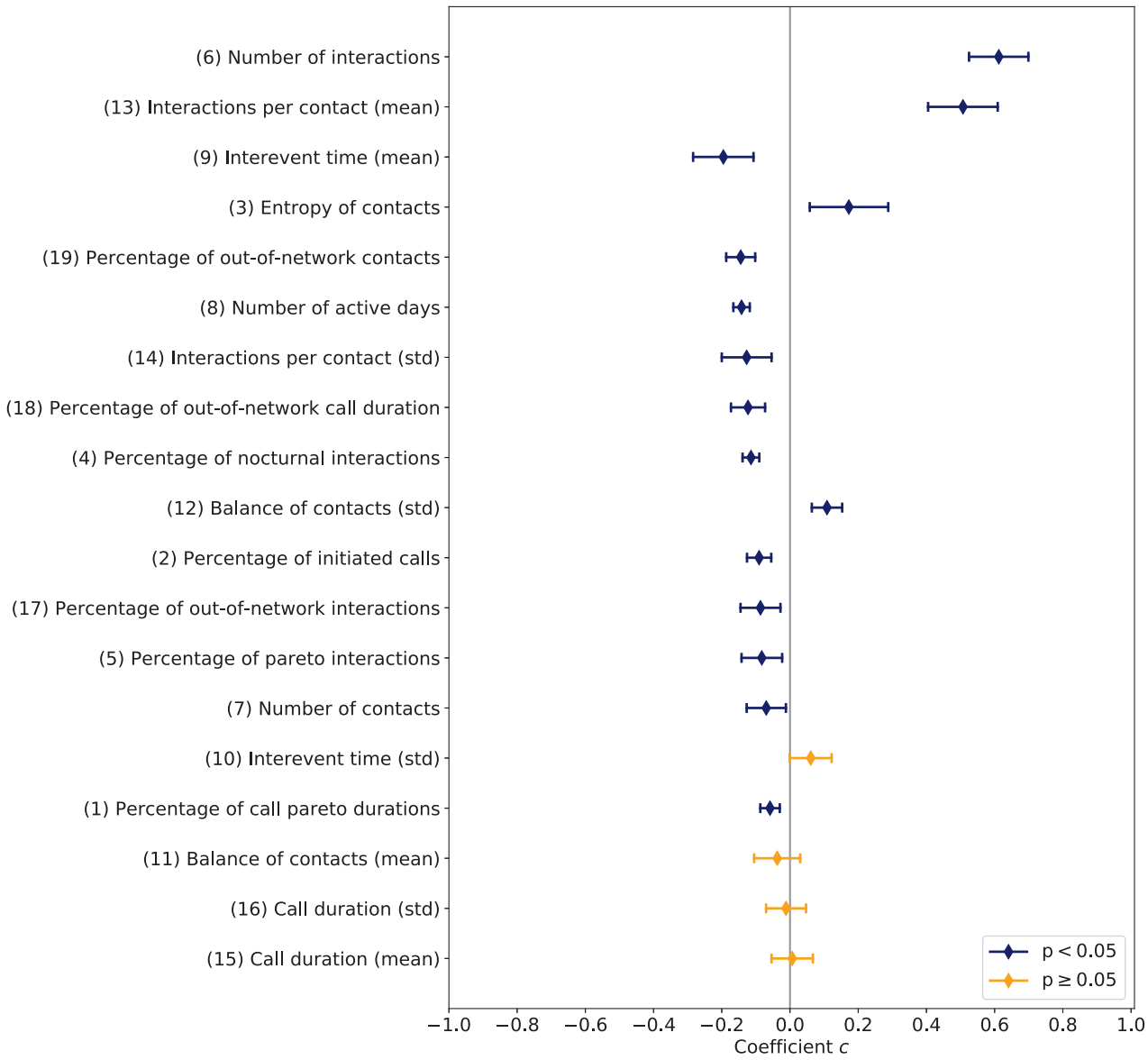


Figure 3.9: **Coefficients of a Logistic Regression for individual identifiability.** For each feature, we plot the coefficient c (with the 95% confidence interval) of a logistic regression classifier with whether a person is more or less identifiable than expected as dependent variable. Features are ordered decreasingly from top to bottom according to the absolute value of c. When holding all other features constant, these results suggest that having more interactions and a well-balanced interaction graph makes individuals more identifiable.

actions, or a well-balanced interaction graph are more identifiable. The largest coefficients c (in absolute value), both as individual predictors (see Fig. 3.10) and taken together, are the number of interactions, the mean number of interactions per contact ($c > 0$), and the mean interevent time, i.e., time elapsed between consecutive interactions ($c < 0$). Interestingly, a person's call duration (both mean and standard deviation) seems to have no impact ($p \geq 0.05$) on identifiability. While the standard deviations of all summary distributions are highly correlated

with their mean ($\rho > 0.7$, see Fig. 3.11), they can still be informative even when other features are accounted for, e.g., the standard deviation of the number of interactions per contact. Last, we note that all other features being the same, the lower a person's number of active days, the more likely they are to be identified, with similar findings for the percentage of nocturnal or out-of-network activity. Finally, we report in Table. 3.3 the p-values for the coefficients of the logistic regression models (with null hypothesis $c = 0$). While our findings suggest the possible influence of the various behavioural features on identification, a causal analysis is beyond the scope of this work.

Table 3.3: **Logistic Regression summary**. For each feature, we report the p-values when all features are used to fit the model **(I)**, respectively when each feature is used as sole predictor variable **(II)**.

	Feature	p**-value (I)**	p**-value (II)**
1	Percentage of call pareto durations	0.000	0.000
2	Percentage of initiated calls	0.000	0.000
3	Entropy of contacts	0.003	0.000
4	Percentage of nocturnal interactions	0.000	0.000
5	Percentage of pareto interactions	0.007	0.000
6	Number of interactions	0.000	0.000
7	Number of contacts	0.017	0.000
8	Number of active days	0.000	0.000
9	Interevent time (mean)	0.000	0.000
10	Interevent time (std)	0.052	0.000
11	Balance of contacts (mean)	0.279	0.000
12	Balance of contacts (std)	0.000	0.000
13	Interactions per contact (mean)	0.000	0.000
14	Interactions per contact (std)	0.001	0.000
15	Call duration (mean)	0.837	0.262
16	Call duration (std)	0.698	0.758
17	Percentage of out-of-network interactions	0.004	0.000
18	Percentage of out-of-network call duration	0.000	0.000
19	Percentage of out-of-network contacts	0.000	0.000

3.5.6 Comparison with prior work and baselines

We compare our attack to previous work from 2014 [297], the only attack in the literature developed for user linkage across call graphs in the context of the D4D challenges (hereafter:

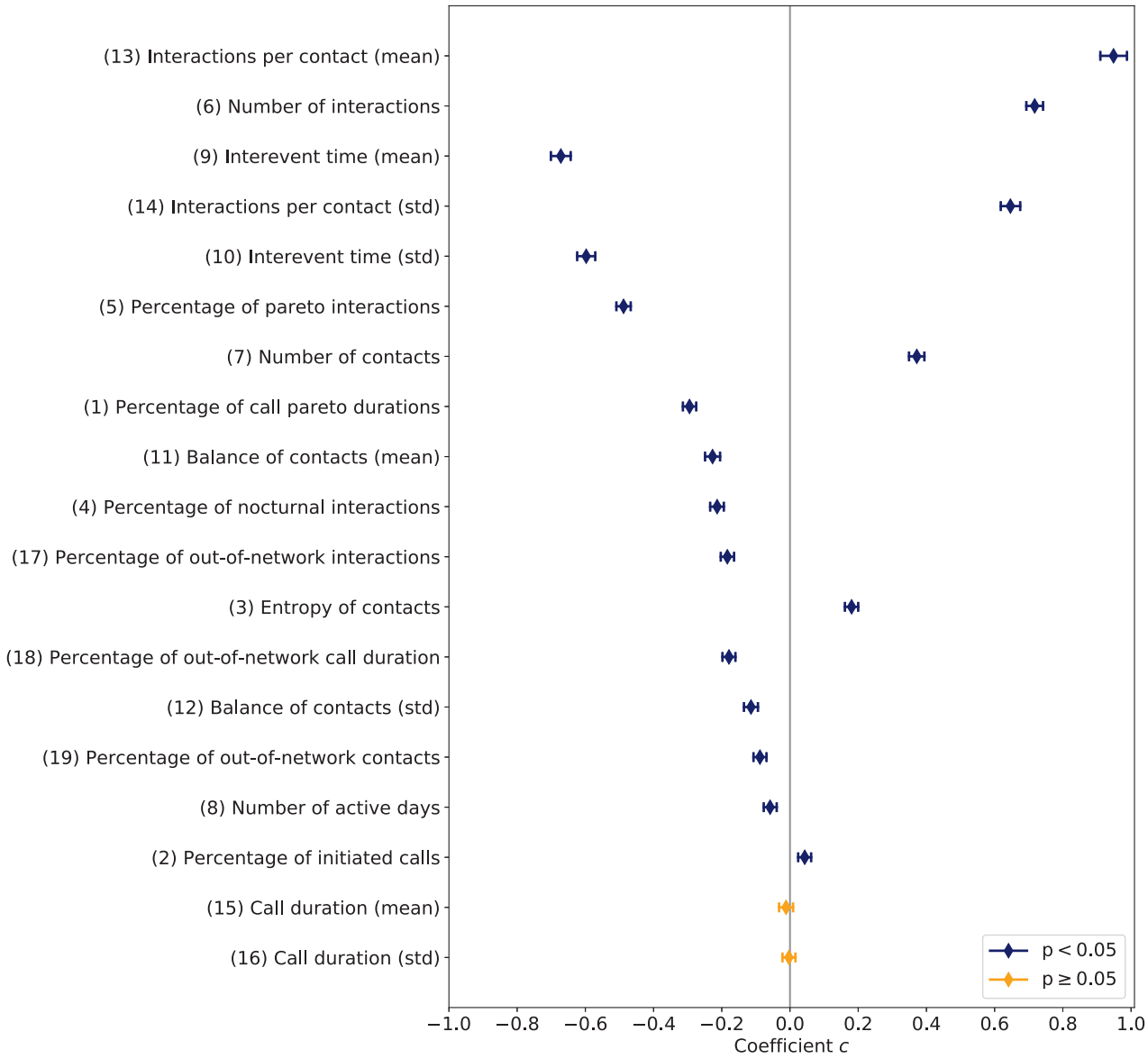


Figure 3.10: **Coefficients of a Logistic Regression for individual identifiability using each feature as sole predictor variable.** For each feature, we plot the coefficient c (with the 95% confidence interval) of a logistic regression classifier with whether a person is more or less identifiable than expected as target variable, and the feature as predictor variable. Features are ordered decreasingly from top to bottom according to the absolute value of c.

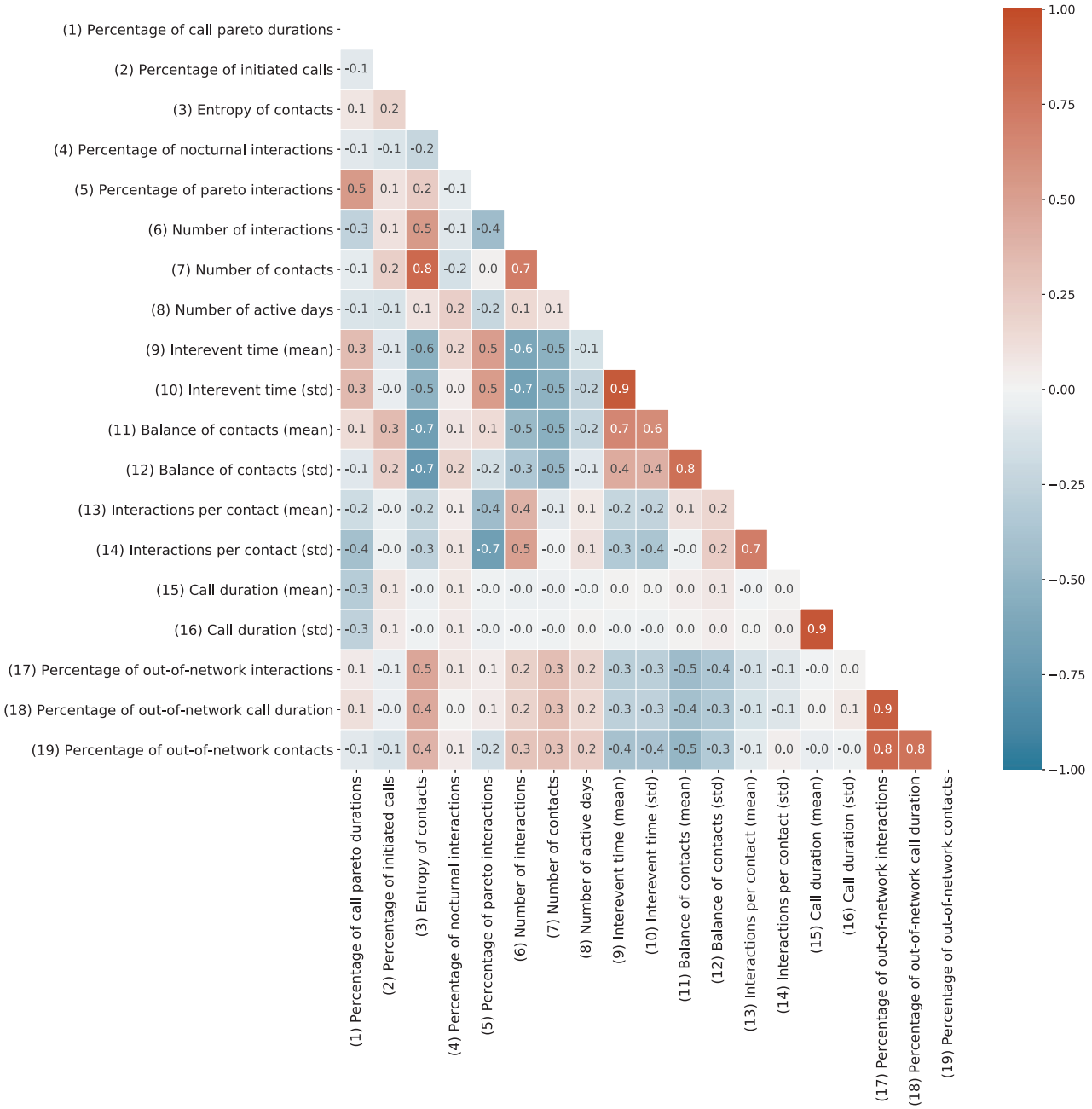


Figure 3.11: **Pairwise feature correlations.** For each pair of features, we compute the Pearson correlation coefficient, rounded to single decimal precision.

ShDa). This work frames re-identification as a binary classification task whose goal is to infer if two nodes in different graphs belong to the same entity. The method uses a random forest classifier trained on hand-engineered features representative of the neighbourhoods of the two nodes in their respective graphs. More specifically, given a node n, the features $\mathbf{h}_n^1$ extracted are the histogram of the degrees of the node's 1-hop neighbours. For a pair of nodes (n, m) with respective features $(\mathbf{h}_n, \mathbf{h}_m) \in \mathbb{R}^F \times \mathbb{R}^F$, the approach constructs a number of F^2 features

of the form:

$$(i,j) \rightarrow \frac{|\mathbf{h}_{ni} - \mathbf{h}_{nj}|}{\max(1, \max(\mathbf{h}_{ni}, \mathbf{h}_{nj}))} \text{ for } 1 \leq i \leq F \text{ and } 1 \leq j \leq F \tag{3.9}$$

We re-implement their approach using the out degree (number of contacts) of a node's neighbours in the k-IIG originating at the node. We use $F = 21$ bins: 20 bins of size 5 and one last bin for degrees between 100 and 1000. When the degrees of 2-hop neighbours are available (i.e., for a 3-IIG), we compute an additional feature vector, $\mathbf{h}_n^2$, consisting of the histogram of the degrees of the node's 2-hop neighbours. We use the random forest implementation of Scikit-learn 0.22.1.

We take the following steps to ensure a fair comparison with our method:

- We convert our attack into a binary classifier over pairs of nodes, using as threshold the inverse distance between the embedding of two nodes: $\frac{1}{1+||e(a)-e(b)||_2}$, where $e : a \rightarrow e(a)$ denotes an embedding method.

- We additionally construct a variant of the ShDa approach that additionally exploits our behavioural features extracted using bandicoot [93]. We concatenate to the node pair features of Eq. 3.9 a new set of features measuring differences in behavioural features of the pair nodes: $(|b_i - b'_i|)_{1 \leq i \leq F}$, where $(b_i)_{1 \leq i \leq F}$ and $(b'_i)_{1 \leq i \leq F}$ denote the behavioural features of i and j. We therefore implement 4 baselines: random forest for $k = 2$ and $k = 3$, without (ShDa) or with behavioural features (ShDa + BF).

- We perform extensive hyperparameter tuning on the training data, selecting the best hyperparameters using 7-fold cross-validation as we did for our model. More specifically, we use $\frac{T'-1}{2} = 7$ disjoint test folds defined as $\{(\mathcal{W}_{2i+1}, \mathcal{W}_{2i+2}) \ : \ 0 \leq i < 7\}$. In each fold, the remaining data is used for training. We do grid hyperparameter search on the random forest parameters `num_trees` $\in \{100, 400\}$, `max_samples` $\in \{10000, 100000\}$ and on using either all the training data or only the two weeks before the test weeks (modulo $T - 1$). We add a ninth hyperparameter resembling the most the original settings of the attack [297], with `num_trees` $= 400$, `min_samples_split` $= 0.1$, `max_features` $= 0.05$ and `max_samples` $= 10000$. The final model is a random forest trained using the best

hyperparameters on the first 14 weeks of data. We report results on N positive pairs and N random negative pairs from the reference and target weeks, $\mathcal{W}_{15}$ and $\mathcal{W}_{16}$, respectively.

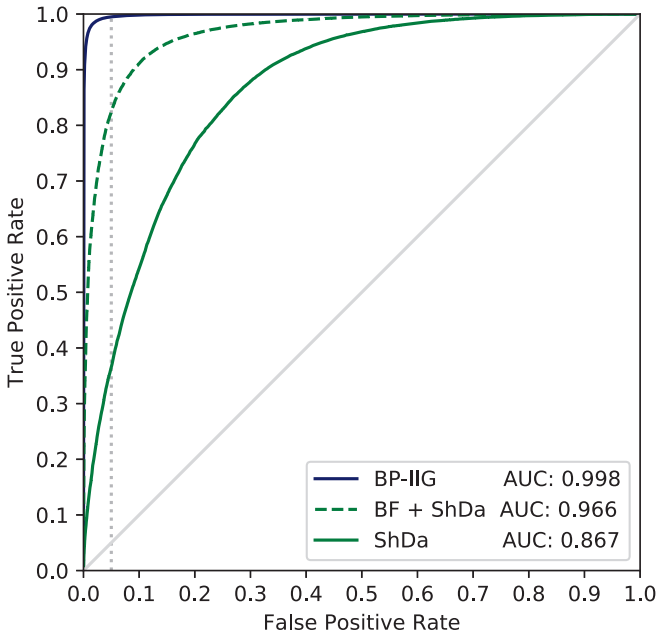


Figure 3.12: **Receiving Operator Characteristic (ROC) curves with Area Under the Curve (AUC) score for the node pair classification task for** $k = 2$. The performance of a random classifier is shown in solid gray.

Fig. 3.12 shows that our approach (BP-IIG, blue line) vastly outperforms previous work (ShDa, solid green line) making profiling attacks a real risk. We report the Receiver Operator Characteristic (ROC) curve and Area Under the Curve (AUC) on the binary classification task for $k = 2$, showing show our approach achieves, on their task and for a false positive rate of 0.05, a true positive rate of 0.99 (AUC=0.998) versus 0.36 for ShDa (AUC=0.868). Our method still outperforms ShDa when we add to it our behavioural features (ShDA + BF, green dashed line) which results in a true positive rate of 0.82 for a false positive rate of 0.05. We refer the reader to Fig. 3.13 for results for other values of k. More importantly, Fig. 3.14 shows how our approach strongly outperforms ShDa on the task of interest: the probability p_k of correctly identifying a person. Here, ShDa alone only achieves a $p_{k=2} = 0.3\%$ versus $p_{k=2} = 52.4\%$ for our attack. Even the improved version, ShDa + BF, only achieves a low $p_{k=2} = 8.3\%$.

Our embedding-based approach further improves on other baselines based on hand-engineered

features, in both the binary classification (Fig. 3.13) and rank-R identification (Fig. 3.14) scenarios. We implement the following baselines:

- An approach that uses the histogram of degrees $\mathbf{h}_n^1$ for $k = 2$ ($[\mathbf{h}_n^1, \mathbf{h}_n^2]$ for $k = 3$), and the $\mathbb{L}_2$ distance for identification. We denote it Degree Features (DF) + $\mathbb{L}_2$.

- An approach that uses the histogram of degrees computed from the number of interactions instead of the number of contacts, and the $\mathbb{L}_2$ distance for identification. We denote it All Interaction Features (AIF) + $\mathbb{L}_2$.

- An approach that uses behavioural features (BF) to represent the k-IIG and the $\mathbb{L}_2$ distance for identification. The approach takes a different form depending on the value of k. We first consider a setup ($k = 0$) where only a node's interaction list is available to the adversary, but not the contacts' identities. In this scenario, the adversary can exploit an individual's interaction patterns but not the 1-hop graph structure. We propose a baseline that uses only the behavioural features without graph information (i.e., features 1-2, 4, 6, 9-13, and 18-19 in Table 3.1, totalling 11 features). We denote this baseline NG-BF + $\mathbb{L}_2$. For $k = 1$, the approach uses all the behavioural features and is denoted BF + $\mathbb{L}_2$. For $k = 2$, the approach concatenates the behavioural features of a node to those of the top 5 neighbours according to the number of interactions, with tie-breaks decided by the total call duration. We denote this approach by BF-N + $\mathbb{L}_2$. In all cases, we standardise the behavioural features using estimates from the training weeks.

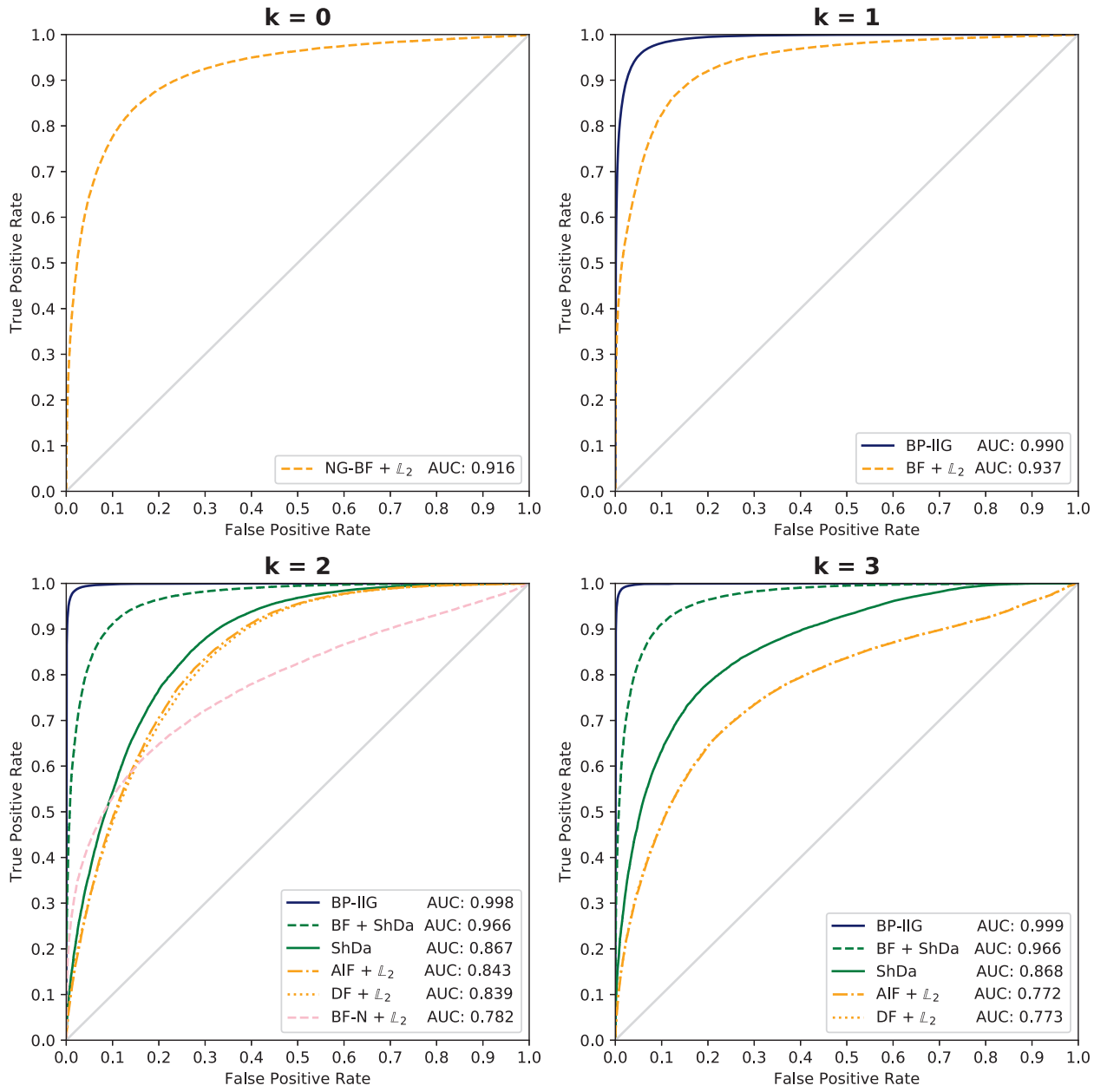


Figure 3.13: **Receiving Operator Characteristic (ROC) curve with Area Under the Curve (AUC) score for the node pair binary classification task**. The performance of a random classifier is shown in solid gray. For each $k \in \{1, 2, 3\}$, our method outperforms the baselines.

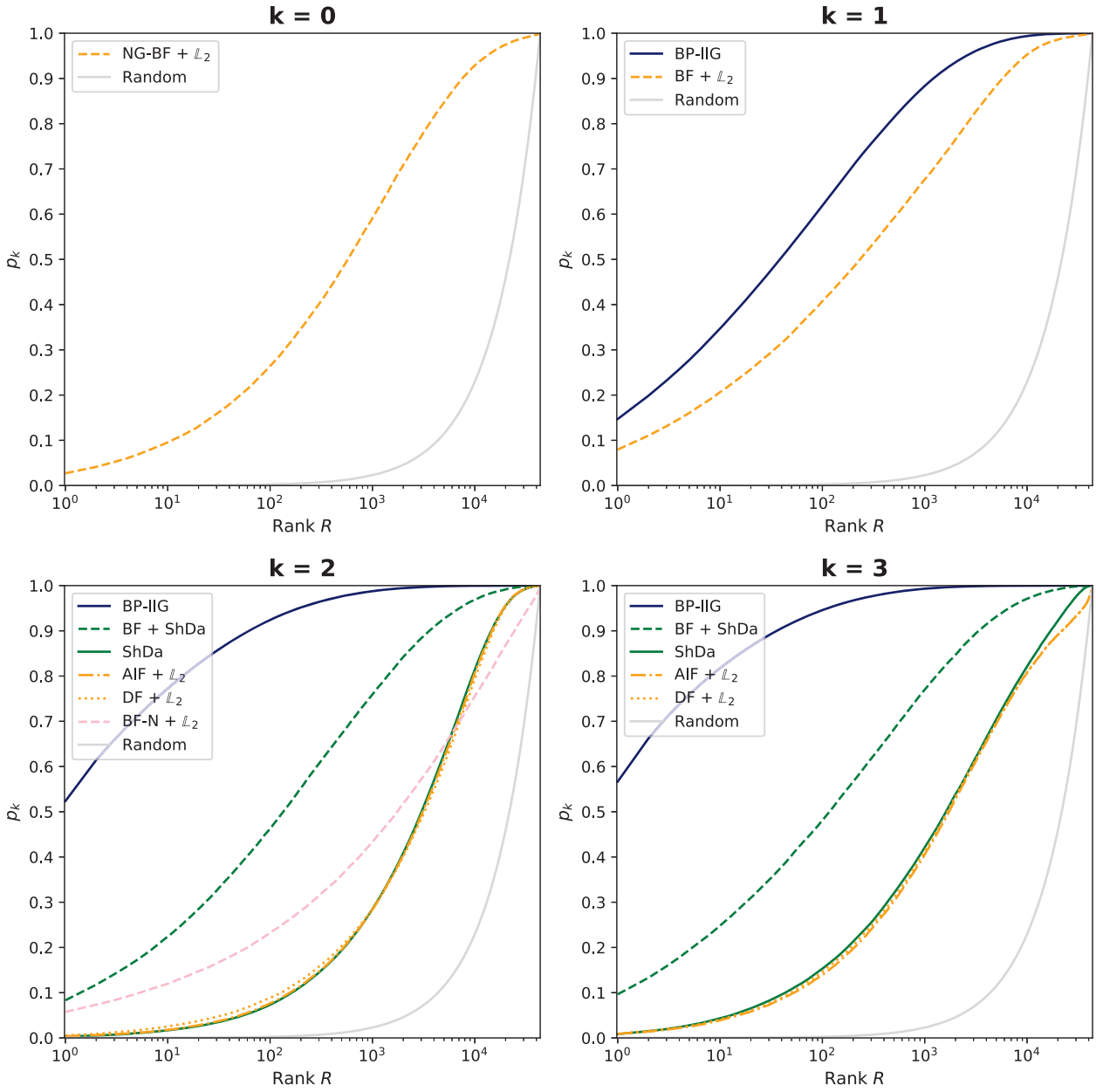


Figure 3.14: p_k, **the probability of identification within rank** R **when the time gap is one week,** $R \in \{1, \ldots, N\}$**.** For each $k \in \{1, 2, 3\}$, our method outperforms all the other approaches.

3.6 Bluetooth close-proximity data

To prevent the spread of COVID-19, governments and companies around the world have been developing and releasing a number of contact tracing apps. Contact tracing apps use Bluetooth to collect close-proximity data between users. If a user becomes infected, they upload to a server data allowing their contacts to be informed that they might have been infected. In the centralised model, application users typically upload the temporary pseudonyms of their contacts [237, 162]. In the decentralised model, they upload data about themselves, typically cryptographic keys, which their contacts can use to deduce that they might have been infected [250, 331, 18, 51]. In another ("hybrid") system, users upload their encounter keys (corresponding to a pair of user identifiers) instead [58]. Numerous application designs based on these protocols have been proposed and are under active development.

In this section, we extend our approach to Bluetooth close-proximity data and show, to the best of our knowledge, for the first time how mitigation strategies relying on changing pseudonyms of both the person and of all of their contacts could fail to adequately protect people's privacy. While our attack does not target a specific application, protocol, or type of protocol (centralised, decentralised, or hybrid), it could form an effective basis for an attack against any system where an adversary has access to a user's social graph over two or more time periods. This could be by design in a centralised system (e.g., the UK's NHSX app reportedly plans to change keys every 24h [237]) or the results of extra data collection in a decentralised system (e.g., the Belgian system reportedly collects the number of encounters with infected users and, for each encounter, the number of days elapsed since the reported contamination of the other user [339]). While the specifications for the reporting of data for epidemiological purposes are currently under discussion, they are likely to include part or all of the infected user's social graph.

We evaluate the effectiveness of our attack using a real-world Bluetooth close-proximity network of 587 university students over 4 weeks [285]. The interaction data consist of the identifiers of the parties, the interaction timestamp and the Received Signal Strength Indication (RSSI), a proxy for the distance between devices. This is the data typically captured by contact tracing apps [237].

Setup. In this experiment, we assume that the adversary has access to (1) the anonymous dataset $\mathcal{D}$ of 1-IIGs from all participants from the first three weeks $\mathcal{T}_1 = \mathcal{W}_{1:3}$ and (2) auxiliary information about a target individual consisting of their 1-IIG from the fourth week $\mathcal{T}_2 = \mathcal{W}_4$. The adversary uses the first two weeks of data $\mathcal{W}_{1:2}$ for training and the second and third weeks $\mathcal{W}_2$ and $\mathcal{W}_3$ for validation. We report the identification results on reference and target weeks $\mathcal{W}_3$ and $\mathcal{W}_4$. By design, the model is neither trained nor validated on data from the target week $\mathcal{W}_4$.

Features. Each interaction in the dataset (a, b, t, r) consists of the identifiers of the parties, a and b, the interaction timestamp t, and the RSSI r. When b is non-negative, it is an identifier for a participant in the study and we map the interaction to two edges, (a, b, t, r) and (b, a, t, r). When $b = -1$, the interaction represents an empty scan and we map it to a single edge $(a, -1, t, r)$. Finally, when $b = -2$, it is an identifier of a device outside the study (henceforth denoted out-of-network), and we map the interaction to a single edge $(a, -2, t, r)$.

We extract 14 behavioural features from every 1-IIG using the bandicoot tool [93]. To these we add 2 other features: the number of empty scans and percentage of out-of-network interactions. Table 3.4 lists the 16 features used and, where applicable, the bandicoot methods used to compute each individual feature. Note that the original dataset records the number of seconds elapsed since a reference timestamp set to 0 instead of the exact timestamp of an interaction. Furthermore, the timestamps are rounded to multiples of 300 (i.e., 5 minutes). As bandicoot expects a real timestamp as input, we replace the lowest timestamp with a timestamp corresponding to 00:00 AM on a Sunday, based on a mention by the authors that the dataset started on a Sunday [285]. We then update the timestamps of all interactions accordingly. In practice, the timestamp only impacts one feature: the percentage of nocturnal interactions (feature 5 in Table 3.4). We acknowledge that the timestamp needed to compute this feature might be inaccurate and leave its estimation for future work. Furthermore, as bandicoot was originally designed for mobile interaction data, it expects an type and direction for each interaction. We here set the type and direction fields to dummy values `call` and `out`, respectively.

Feature transformation. We transform the features 7-12 having skewness coefficient ≥ 3

Table 3.4: **The list of the 16 features used for the Bluetooth close-proximity interaction data.** We also list the corresponding methods to compute the features using the bandicoot toolbox. A dash symbol (-) indicates that the feature was not computed using bandicoot.

	Feature description	Bandicoot method
1	Number of days, between 1 and 7, when the user was active	`active_days`
2	Number of contacts	`number_of_contacts`
3-4	Mean/standard deviation of the negative RSSI of the user's interactions	`call_duration`
5	Percentage of interactions that occurred between 7PM and 7AM	`percent_nocturnal`
6	Entropy of contacts	`entropy_of_contacts`
7-8	Mean/standard deviation of the balance of the user's contacts, where the balance is the ratio between the number of outgoing interactions with a contact and the total number of interactions.	`balance_of_contacts`
9-10	Mean/standard deviation of the time between two consecutive interactions recorded for the user	`interevent_time`
11-12	Mean/standard deviation of the number of interactions with each contact	`interactions_per_contacts`
13	Percent of contacts that account for 80% of the user's interactions	`percent_pareto_interactions`
14	Number of interactions	`number_of_interactions`
15	Number of empty scans (i.e., where the other node is -1)	-
16	Percentage of out-of-network calls (i.e., where the other nodes is -2)	-

(computed from weeks $\mathcal{W}_1$ and $\mathcal{W}_2$) by fitting a Box Cox transformation after adding 1 to each feature for it to be strictly positive. The transformation is fitted on data from nodes in the training dataset. We then standardise all features. All the transformations are fitted on the training data.

Training architecture. As embedding model for the 1-IIGs, we use an MLP with one hidden layer of size 64 and output size of 25. The other training details are the same as Sec. 3.4.2, except that we perform hyperparameter search on different values for the batch size $B \in \{16, 32, 64\}$, the learning rate $\eta \in \{0.01, 0.05, 0.1\}$ and the weight decay parameter $\alpha \in \{0.001, 0.01\}$.

Training details. As only two weeks of data are available for training the model, we use

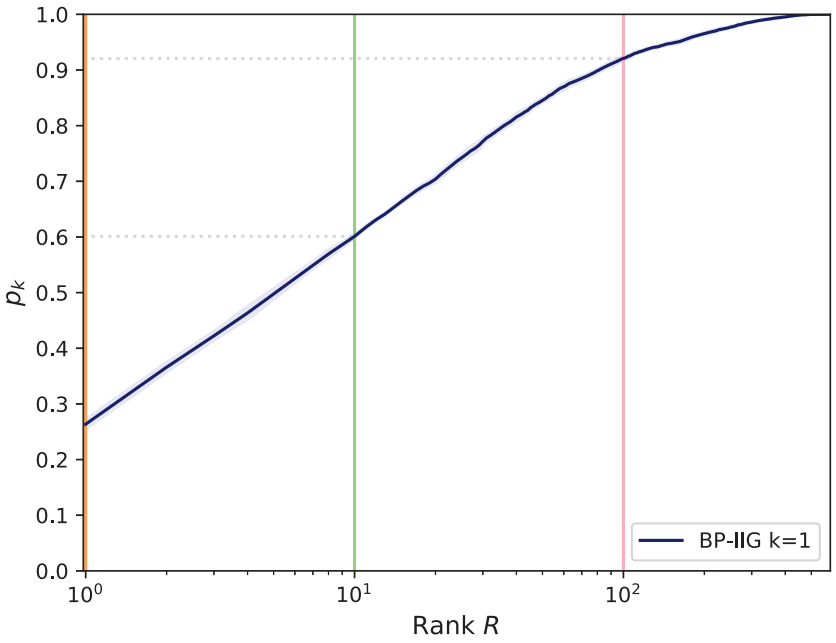


Figure 3.15: **Probability of identification in a bluetooth close-proximity network.** We plot $p_{k=1}$, the probability of identification within rank R for $k = 1$. The 95% confidence interval is shown in light blue. Our method correctly identifies people $p_{k=1} = 26.4\%$ of the time based on their 1-IIGs. It is able to identify the right person $p_{k=1} = 60.1\%$ of the time among the top 10 candidates.

data augmentation to increase the number of samples of weekly behaviour. More specifically, we generate 8 consecutive weeks of data, each time shifting the starting day by one day. Formally, denoting by $d_1, \ldots, d_{14}$ the 14 days of training data, we generate 8 overlapping weeks $\mathcal{W}'_1, \ldots, \mathcal{W}'_8$, with $\mathcal{W}'_i = d_i \cup \ldots \cup d_{i+6}$. We keep all nodes with at least one interaction in all of the weeks. This results in $N = 587$ people on which we report results, out of the 700 available in the original dataset.

Results. Fig. 3.15 shows that for $k = 1$ our approach is able to identify target individuals $p_{k=1} = 26.4\%$ of the time among the 587 people. Out of 10 people ($R = 10$), it is able to identify the right person $p_{k=1} = 60.1\%$ of the time. While our dataset is too small to evaluate for larger values of k, we expect the results to further increase when more information is available.

3.7 Discussion

We now discuss some of our assumptions on the attack model and explore alternative approaches. All the experiments described in this section use the mobile phone interaction dataset.

Applicability to re-pseudonymisation. Embedding models are generic feature extractors, i.e., a function taking as input a k-IIG and extracting a set of features, where the model parameters are trained such that the features are stable over time. In particular, once trained this function can be applied to new k-IIGs, including from individuals different from the ones in the model's training dataset. As our attack works by computing the Euclidean distance between a target's embedding and a candidate set of reference embeddings, only one reference sample of an individual is enough to perform classification. We argue that this makes the model applicable to evaluating the robustness of weekly re-pseudonymisation. Indeed, an adversary could potentially train an embedding model on a publicly available dataset, or a dataset available to them, and use it to re-identify new users in other interaction datasets.

To assess how well the model performs when applied to new populations, we examine its performance on three different scenarios: when testing is performed on a set disjoint from the training set in the identities of the individuals, time periods used, or both, as illustrated in Fig. 3.16. More specifically, we use the first $T'+1 = 16$ weeks of data and all k-IIGs to design the following experiment. We sample the disjoint validation and test usersets $\mathcal{I}_{\text{val}}$ and $\mathcal{I}_{\text{test}}$ of size $N/4$ from $\mathcal{I}''$ uniformly at random without replacement. We use as validation time period the reference week $T'-3$ and target week $T'-2$, and as test time period is the reference week $T'-1$ and target week T. For a fair comparison across scenarios, we always select hyperparameters (and early stop the models) on the validation data and report the identification performance on the test data. We consider three different scenarios for the training data:

- In the first scenario (Fig. 3.16b), we use $\mathcal{I}_{\text{val}} \cup \mathcal{I}_{\text{test}}$ as the training userset and the first $T'-4$ weeks as the training time period . This scenario is similar to our main experiment (Fig. 3.16a), as data from the test users (in a disjoint time period) is seen during training.

Table 3.5: **Probability of identification within rank 1 for different scenarios and values of k.**

	First scenario	Second scenario	Third scenario
$k = 1$	22.8	23.1	23.0
$k = 2$	61.5	62.2	60.5
$k = 3$	66.5	66.8	66.6

- In the second scenario (Fig. 3.16c), we use the remaining users $\mathcal{I}' \setminus (\mathcal{I}_{\text{val}} \cup \mathcal{I}_{\text{test}})$ as the training userset and all the available weeks as the training time period. In the graph neural network literature, this scenario is referred to as *transductive* because of the potential overlap between training and test data. The test k-IIGs could indeed partially overlap with the training k-IIGs. For example, Alice, the originating individual for a k-IIG in the training set, could be a neighbour of Bob, the originating individual for a k-IIG in the testing set.

- In the third scenario (Fig. 3.16d), we use the remaining users $\mathcal{I}' \setminus (\mathcal{I}_{\text{val}} \cup \mathcal{I}_{\text{test}})$ as the training userset and the first $T' - 4$ weeks as the training time period.

We train the models using the same architecture, hyperparameters, and training details (and select the best learning rate among the same set of possible values) as the main experiment.

Table 3.5 shows that our attack performs similarly across the three scenarios for all values of $k \in \{1, 2, 3\}$. Our attack is equally able to identify people unseen during training in time periods also unseen during training ($p_{k=2} = 61.5\%$) as in cases when the same people ($p_{k=2} = 62.2\%$) or time periods ($p_{k=2} = 60.5\%$) used in testing are seen during training. We observe similar results for $k = 1$ and $k = 3$.

Past auxiliary information. While the attack model is general (see Sec. 3.2), we have throughout the work assumed that the auxiliary information comes from a time period posterior to the dataset $\mathcal{D}$ ($t'_1 < t'_2$). Using our approach on $k = 2$ and the last 18 weeks of data, we compared the performance of a model trained on first 9 weeks and tested on the following 9 weeks, with that of a model trained on the last 9 weeks and tested on the first 9 weeks. The two models gave the same performance ($p = 0.58$). This confirms the generality of our model.

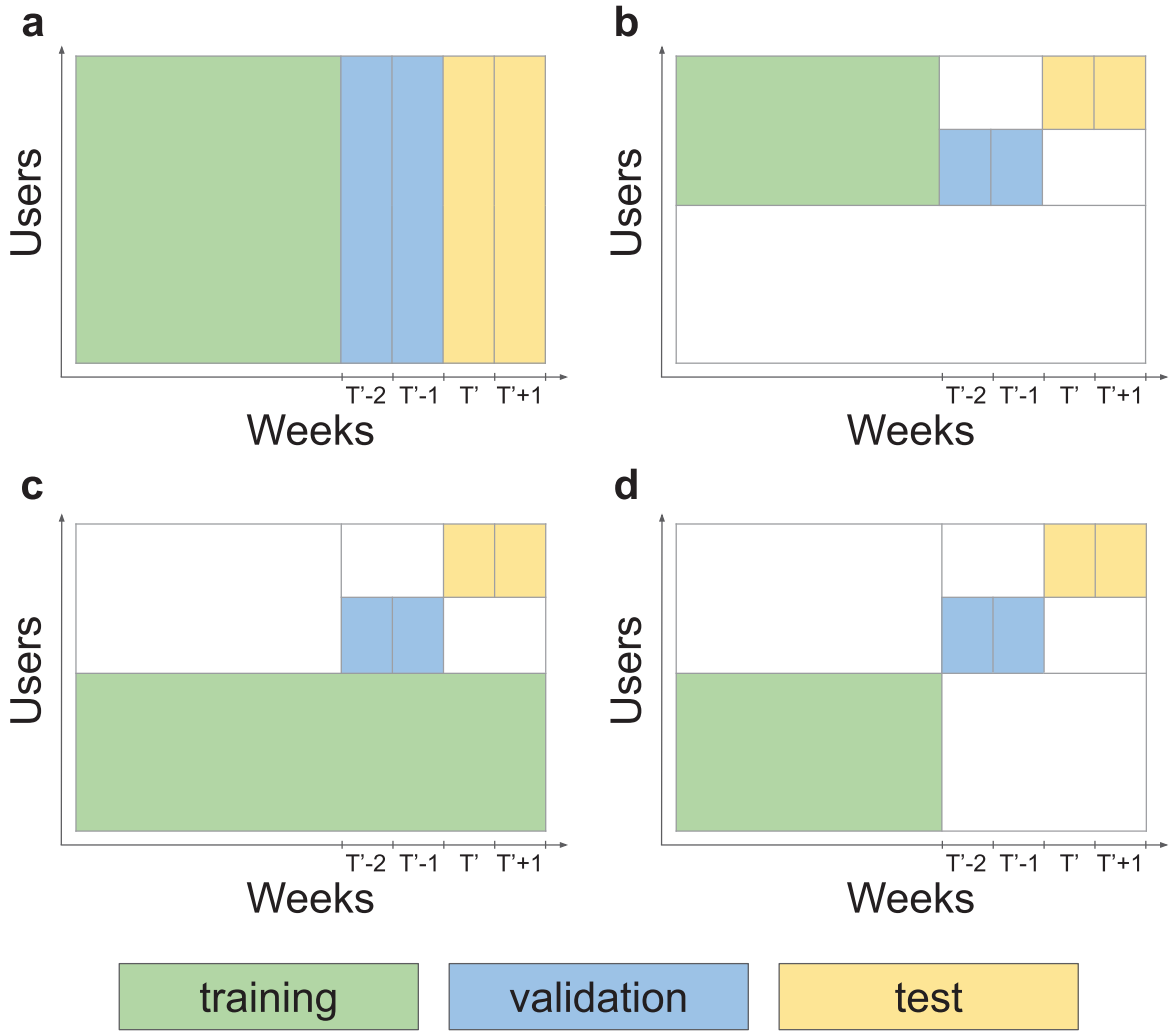


Figure 3.16: **Illustration of data splits in different evaluation scenarios**. We perform testing, validation and training on sets disjoint in the time periods (**a** and **b**), the identities of the originating individuals of the k-IIGs (**c**) or the time periods and the identities of the originating individuals of the k-IIGs (**d**). We use the green, blue and yellow splits for training, validation and testing, respectively. To compute validation and test metrics, we use the first week in the split as reference week and the second week as target week. Scenario **a** is used for the main experiment of the work, while the others are used to evaluate the generalisation ability of our attack.

Consistency of neighbours' pseudonyms across weeks. While we assume, again in line with previous practices [297, 39], that in the dataset $\mathcal{D}$ pseudonyms are identical over time for nodes in k-IIGs of the same individual, this is not a requirement of our approach. Re-pseudonymisation of nodes over time might be used, for example, to avoid direct access to an individual's interactions over a long period of time. Our approach would still work even if the dataset $\mathcal{D}$ consisted of weekly k-IIGs with different pseudonyms for the same node appearing

in two weekly k-IIGs of the same person, so long as the adversary knows the identity of the originating individual in each weekly k-IIG. Indeed, our graph aggregation functions (bandicoot feature extraction and graph neural network) are agnostic to the identities of neighbours that need to be consistent only within the week we aggregate over.

Alternative architecture. As an alternative to our attention-based approach, we also implemented a Message Passing Neural Network (MPNN) architecture [135]. For the MPNN architecture, the message between a node and a neighbour is computed by applying a linear layer followed by a ReLU nonlinearity to the concatenation of their features. It achieves performance similar to our method (53.5% vs 53.0% on the validation set, $k = 2$, 10 runs, p-value: 0.11, 95% confidence intervals: [52.6, 53.4] for the MPNN and [53.0, 53.9] for our approach). We thus opt for the attention-based approach, which is simple and more interpretable.

Analysis of attention weights. To gain insights into the graph neural network's capacity to assign different weights to different neighbours of a node in the aggregation step, we analyse for $k = 2$ the attention weights between the N individuals and their neighbours in the reference week T'. We compute two metrics capturing the variation between weights assigned to different neighbours. The first metric is the normalised entropy of the attention vector in each aggregation layer $s = 1, \ldots, 2$. This metric takes as input a vector with positive entries that sum to 1, $(w_1, \ldots, w_{|\mathcal{N}(i)|})$. It computes the ratio between the entropy $\sum_{1 \leq q \leq |\mathcal{N}(i)|} -w_q \ln(w_q)$ of the vector and $\ln(|\mathcal{N}(i)|)$, the maximum entropy for a discrete probability distribution with $|\mathcal{N}(i)|$ possible values. The second metric is the normalised range of a vector, defined as $\mathbf{w} \to (\max(\mathbf{w}) - \min(\mathbf{w}))/\min(\mathbf{w})$. This metric captures the relative gap between the largest and smallest attention weight.

Fig. 3.17a shows the cumulative distribution of the normalised entropy of the attention weight vectors, and Table 3.6 shows three random samples for various entropy intervals. While entropy values tend to be high, there is sufficient and meaningful variation in the attention weights, which suggests that the model is able to capture the relative importance of neighbours. This is further supported by Fig. 3.17b which shows the cumulative distribution of the normalised range of a vector. In more than 90% of the cases, $\max(\mathbf{w}) - \min(\mathbf{w}) \geq \min(\mathbf{w})$, i.e., $\max(\mathbf{w}) \geq$

$2\min(\mathbf{w})$ which means that the largest attention weight is at least twice as large as the smallest one(s). This further shows how attention weights help discriminate between the neighbours.

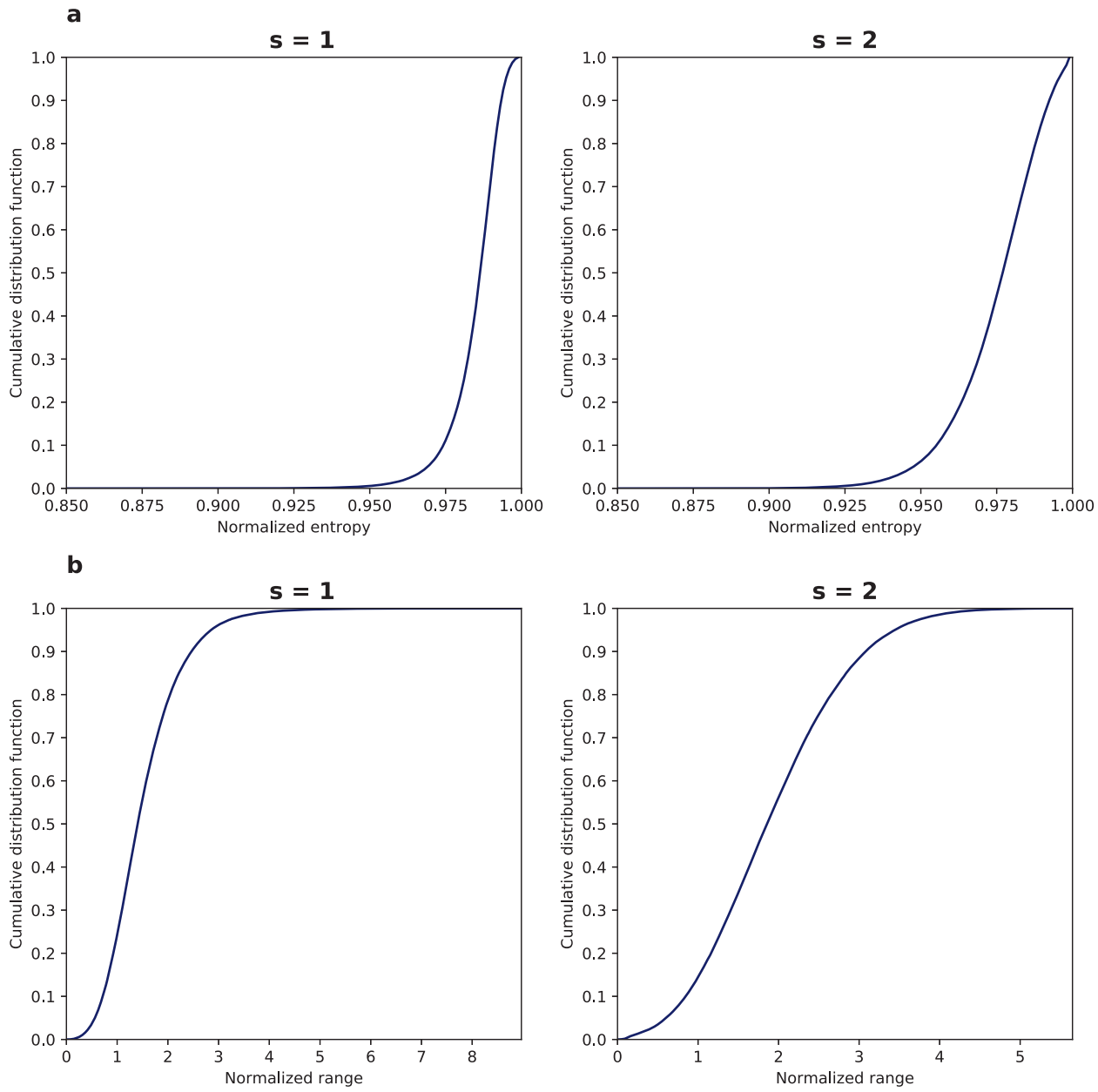


Figure 3.17: **Analysis of attention weights.** We report the cumulative distribution function of the **a** normalised entropy and **b** normalised range of the attention weights used to aggregate the neighbours the $N = 43,606$ individuals. The results are computed over 2-IIGs from the reference week and aggregation steps $s = 1, 2$.

Table 3.6: **Examples of attention weights for various intervals of the normalised entropy.** We randomly select one attention weight vector in each interval ($s = 1$). In all cases, one or two neighbours have an attention weight at least twice as large as the lowest weight(s).

Interval	Normalised entropy	Attention weight (sorted)
[0.85, 0.90)	0.8990	[0.26, 0.24, 0.19, 0.17, 0.06, 0.04, 0.04]
[0.90, 0.95)	0.9495	[0.26, 0.19, 0.11, 0.10, 0.09, 0.09, 0.08, 0.08]
[0.95, 1.00]	0.9827	[0.24, 0.14, 0.13, 0.13, 0.12, 0.12, 0.12]

3.8 Conclusion

In this work, we propose a new behavioural profiling attack model exploiting the stability over time of people's k-hop interaction networks. We evaluate its effectiveness on two real-world offline and online interaction datasets and show the risk of identification to be high.

Our results provide strong evidence of the urgent need to consider profiling attacks when evaluating whether systems, protocols, or datasets satisfy the Article 29 WP's definition of anonymisation [19]. In particular, they show how people's interaction patterns online and offline remain identifiable across long periods of time allowing an adversary to link together data coming from disjoint time periods with high accuracy even in large datasets. Our results challenge current data retention practices and, in the context of the recent COVID-19 pandemic, whether some of the collected data would satisfy the Article 29 linkability criteria. They finally further question the policy relevance of de-identification techniques [263] and emphasise the need to rethink our approaches to safely use non-personal data. In particular, legal and access control mechanisms are necessary to protect data retained in pseudonymised format, and privacy engineering solutions such as query- and question-and-answer-based systems, local DP mechanisms, or secure-multiparty computation could be deployed to help use data anonymously [89].

Chapter 4

Automated attacks against query-based systems

The work presented in this chapter is based on the following peer-reviewed conference article: Cretu, A.-M., Houssiau, F.*, Cully, A., and de Montjoye, Y.-A. (2022). "QuerySnout: Automating the Discovery of Attribute Inference Attacks against Query-Based Systems". In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (pp. 623-637).* `https://dl.acm.org/doi/10.1145/3548606.3560581`

An extended version that includes the appendix is available as an arXiv preprint at `https://arxiv.org/abs/2211.05249`.

Florimond Houssiau and I are co-first authors on this paper. We collaborated in the design of the method, the design of the experiments, and the writing of the paper. I implemented all the methods and experiments and generated all the figures. Florimond Houssiau implemented the query-based systems, designed the extension of the method to budget-based mechanisms, and wrote the proofs relating to differential privacy.

4.1 Introduction

Query-based systems (QBS), controlled interfaces through which analysts can query a database of records, have the potential to enable privacy-preserving anonymous data analysis at scale. As the curator keeps control over the data, they can audit queries sent by analysts and ensure that the answers returned do not reveal individual-level information. Typical queries include histograms, counts, correlations between attributes, and other aggregates over individual records. QBS interfaces can range from an online interface and API [246, 83], to languages such as SQL [213, 123] and the submission of scripts [243, 248].

It has however long been known that only releasing aggregate information is not sufficient to protect privacy. As early as 1979, Denning et al. [97] theorised difference attacks ("trackers") against databases accessible through user-specified counts. Researchers have shown that releasing marginals or simple counts can reveal the presence of a target individual in the private dataset [155, 109]. Famously, Dinur and Nissim proved in 2003 that a database can be reconstructed with good accuracy from a large number of aggregate statistics [101]. This issue is particularly acute for QBSs, where adversaries can design queries to infer information about specific people, including by exploiting vulnerabilities or implementation bugs of the system.

In response to these risks, increasingly sophisticated defence mechanisms have been put in place. These include combining query set size restriction with noise addition mechanisms [123, 246], the use of unbounded static noise [123, 107], the introduction of limits on the number of queries (e.g., as a privacy budget in Differential Privacy [213]), and even online evaluation [225] and rewriting of queries [172]. Computer security tools such as access control, logging queries, code verification, and AI-based anomaly detection mechanisms are typically then deployed on top of QBS-specific mechanisms.

While these measures have helped prevent and mitigate privacy risks [123], the risk of unknown strong "zero-day" attacks has stalled the development and deployment of QBSs. Manually designing and implementing attacks against complex and expressive QBSs is a difficult and painstaking process. It typically requires careful analysis of the system by experts and can

take months. Furthermore, existing attacks have only exploited a small subset of the syntax of modern expressive QBSs. Limiting the risk of existence of strong unmitigated attacks is thus essential to unlock the potential of QBSs and make individual-level data available to researchers and companies while strongly preserving privacy in practice.

Contributions. We here propose QuerySnout, the first method to automatically discover privacy vulnerabilities in query-based systems. We frame the discovery of attacks against a QBS as a black-box optimisation problem. Specifically, we formalise an attack as a multiset of queries jointly with a mathematical rule (e.g., a machine learning model) to combine answers to the queries in order to reveal a particular secret. Given a threat model and black-box access to a QBS, we optimise the attacks with respect to their performance at inferring the secret.

At a high level, QuerySnout analyses the query answering behaviour of the QBS for patterns that can be used to infer the secret. QuerySnout is fully automated, combining (1) evolutionary search techniques to discover the right set of queries to ask with (2) a machine learning model ("rule") trained to infer the secret from query answers. To efficiently explore the search space of possible attacks, we design a novel mutation operator tailored to this problem.

We instantiate our approach on *attribute inference attacks*, where the secret is the sensitive attribute of a target record. We study two attack scenarios. The first scenario (AUXILIARY) assumes that an adversary has access to an auxiliary dataset similar to the private dataset, e.g., drawn from the same distribution. This is a common assumption across the broader literature [264, 302]. The second scenario (EXACT-BUT-ONE) assumes that an adversary has access to the entire private dataset protected by the QBS except for the sensitive attribute. Attack models like this one, relying on a very strong adversary, are used in the literature to evaluate privacy protections in the "worst-case" scenario [163]. They can help uncover flaws in the system design or implementation and audit systems.

We use QuerySnout to attack two real-world query-based systems, Diffix [123] and Table-Builder [246], and a generic query-based system. We show our attack to be highly successful against all three systems when protecting three real-world datasets, matching or outperforming previous expert-designed attacks [67, 277, 129]. A post-hoc analysis of the attacks found by

QuerySnout in the AUXILIARY scenario suggests that they exploit the same vulnerability as the manual attacks, but more effectively. We propose a heuristic to extend our method to budget-based QBSs (such as those guaranteeing differential privacy [107]), and use QuerySnout to attack a simple QBS based on the Laplace mechanism. We show that our attack achieves near-optimal accuracy for $\varepsilon \in \{5, 10\}$.

Finally, we discuss how our method can be extended to other attack models – e.g., to perform membership inference attacks – and larger query syntaxes. Our results suggest that QuerySnout can be used to evaluate the privacy protection offered by QBSs and help improve their design by identifying vulnerabilities.

4.2 Background

4.2.1 Query-based systems

We consider a *data curator* entity, such as a business or a government agency, holding data about a set of users denoted by $U \subset \mathcal{U}$, with $\mathcal{U}$ a population of users. The data consists of values for an ordered set of attributes $\mathcal{A} = (a_1, \ldots, a_n)$, e.g., age and nationality. Each attribute $a \in \mathcal{A}$ has a set $\mathcal{V}_a$ of acceptable values. We denote by *individual record* r^u the data available about a user u, consisting of the corresponding values for attributes in $\mathcal{A}$: $r^u \in \mathcal{V}_{a_1} \times \cdots \times \mathcal{V}_{a_n}$. We denote by $r^u_{\mathcal{A}'}$ the restriction of a record to a subset of attributes $\mathcal{A}' \subset \mathcal{A}$ and by r^u_a the restriction to one attribute $a \in \mathcal{A}$. For instance, if the attributes collected are $\mathcal{A} = (age, nationality)$, a person's record could be (30, Romania). Given a user set U, we call *dataset* the multiset of individual records D for users in U. This means that the same record can appear more than once if different users in U have the same values for these attributes.

We assume that the data curator allows data analysts to access information about the dataset D through a query-based system (QBS). The QBS allows the data analyst to retrieve answers to queries about the dataset without directly accessing the individual records [89]. The interface can use a query language such as SQL or a GUI to define the semantics of queries that can

be asked. Formally, we denote by *query space* $\mathcal{Q}$ the set of queries that can be asked to the system. We denote by $T : \mathcal{D} \times \mathcal{Q} \to \mathbb{R}$ the function that gives the *true answer* to a query over a dataset.

In this work, we focus on tabular datasets having categorical attributes, i.e., $|\mathcal{V}_i| < \infty, i = 1, \ldots, n$. This includes non-ordinal attributes (e.g., gender), discrete ordinal attributes (e.g., age encoded as a round integer) and bucketised continuous attributes (e.g., income brackets).

We assume that the QBS provides answers to, at least, counting queries of the form:

$$\begin{aligned} &\text{SELECT COUNT(*)} \\ &\text{WHERE } a_{i_1}\ c_{i_1}\ v_{i_1} \text{ AND } \ldots \text{ AND } a_{i_{k-1}}\ c_{i_{k-1}}\ v_{i_{k-1}} \text{ AND } a_{i_k}\ c_{i_k}\ v_{i_k}, \end{aligned} \tag{4.1}$$

where $\{a_{i_1}, \ldots, a_{i_k}\} \subset \mathcal{A}$ is a subset of attributes, $v_{i_j} \in \mathcal{V}_{a_j}$ is an admissible value for the i_j-th attribute for $j = 1, \ldots, k$ and c_{i_j} can be either the $=$ (equal to) or $\neq$ (different from) condition. For instance, the query $q =$ SELECT COUNT (*) WHERE $a_1 = 5$ returns the number of users in a dataset D having the first attribute value equal to 5. This is the quantity we denote by the true answer $T(D, q)$. While QBSs may implement other queries as well, such as sums or averages, or allow for logical disjunction (OR) in addition to conjunction (AND) operators, this is a minimal syntax that captures k-way marginal statistics, a popular class of queries [246, 123].

Formally, we consider a query-based system to be a randomised function $R : \mathcal{D} \times \mathcal{Q} \to \mathbb{R}$ assigning to a dataset and query pair $(D, q) \in \mathcal{D} \times \mathcal{Q}$ a real-valued random variable $R(D, q)$. The answers provided by the QBS are usually designed to be *similar* to the true answer ($R(D, q) \approx T(D, q)$) but not equal, in order to protect user privacy. We use the more general notion of a random variable because to preserve the privacy of users, query-based systems commonly implement mechanisms for post-processing that involve randomisation (e.g. noise addition). When the answer is fixed, $R(D, q)$ is a deterministic random variable. Importantly, we assume that when prompted with a query q, the QBS only returns an instance of the random variable $R(D, q)$, but not any additional information, such as the subsets of records selected by the query.

Multiple building blocks are typically combined to preserve privacy. These include noise addition [96] with a range of distributions (e.g. Laplace [107], Gaussian [123], uniform [125]), suppression of answers below a threshold (called *query set size restriction*) [97], and restrictions on the set of allowed queries $\mathcal{Q} \subset \mathcal{Q}'$ or the number of queries that can be meaningfully answered[1] [107].

Most real-world systems are deterministic [246, 123], either through seeded noise or caching, ensuring that the same answer is returned every time the same query is sent. This is desirable from both a privacy perspective (preventing averaging attacks) and a utility perspective (always returning the same answer). In some cases though, systems might be non-deterministic. This can be by design, e.g., lack of awareness of averaging attacks; because of implementation or operational issues, e.g., non-functional cache or answer retrieving mechanism; or in cases where the same dataset is made available from different instances of the QBS seeded differently, e.g., distributed computing.

All the QBSs on which we empirically validate our attacks implement one of these building blocks or combine several of them. We refer the reader to Sec. 4.5.3 and Sec. 4.7.2 for a formal description of these systems. They include both deterministic and non-deterministic systems. Furthermore, they include both ad-hoc anonymisation mechanisms such as TableBuilder [246] and Diffix [123] and budget-based mechanisms implementing differential privacy [107], a formal privacy guarantee described in Sec. 2.2.2.

4.2.2 Evolutionary algorithms

Evolutionary algorithms are a family of optimisation algorithms particularly well suited when the structure of the problem is discrete, unknown or too complex to be described mathematically, and the search space is very large [221]. The goal of an evolutionary algorithm is to produce one or more high-quality *solutions* s to an optimisation problem with search space $\mathcal{S}$, as quantified by a *fitness* function $\mathcal{F} : \mathcal{S} \to \mathbb{R}_+$. The fitness function assigns a score $\mathcal{F}(s)$

[1] Although this mechanism cannot be expressed formally as $R(D, Q)$, since it requires memory of the number of queries performed. We use this notation for simplicity.

to each solution s, measuring its quality for the specific use case. An evolutionary algorithm consists of an iterative procedure operating on a *population* of P solutions $s_1, \ldots, s_P \in \mathcal{S}$. Its goal is to improve the quality of solutions over time using the feedback from the fitness scores of solutions in the population. This improvement is typically achieved by (1) maintaining a population of high-quality solutions selected among those discovered so far, and (2) evolving the population over many iterations (*generations*) to improve their quality. In each generation, new solutions (*offsprings*) are generated by applying small random changes via a *mutation operator* $\mathcal{M} : \mathcal{S} \to \mathcal{S}$ to existing solutions (*parents*), or by combining them via a *crossover operator* $\mathcal{C} : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$. While the ideas behind the algorithm are general, its building blocks: solution, fitness function, and the mutation and crossover operators need to be defined for the specific use case.

4.3 Attack model

We propose a general targeted attack against query-based systems, which we call *automated query discovery attack*. The attack is composed of a search mechanism for a multiset of queries and a rule to combine them. While we focus on attribute inference attacks, our approach can be extended to other attacks, e.g., membership inference attacks.

4.3.1 Adversary access to the query-based system

We assume that the adversary has access to the query-based system protecting the dataset of interest (*target QBS*), in the sense that they can send queries to it. We furthermore assume that the number of queries that can be performed on the target QBS is limited, e.g. to a few hundreds, as queries are typically logged and rate limited.

We also assume that the adversary has black-box access to the QBS software. The QBS software would typically be available freely or for a fee, potentially in compiled form. Alternatively, the adversary might replicate the protection deployed by the target QBS based on public

information. Formally, this means that the adversary can retrieve samples from $R(D', q)$ for any query $q \in \mathcal{Q}$ and dataset $D' \in \mathcal{D}$. Since the adversary chooses D', they also know the true query answer $T(D', q)$ and can leverage this signal to devise attacks. Note that QBSs are typically initiated with a *seed* for pseudo-random noise generation: we assume that each QBS is initiated with a different seed, and that the adversary does not know the seed used in the target QBS.

4.3.2 Attribute inference attack

The adversary's goal is to infer the target user's value for one of the attributes (the *sensitive* attribute) $s = r^u_{a^*},\ a^* \in \mathcal{A} \setminus \mathcal{A}'$. We assume that the adversary knows part of the record of a *target user* $u \in U$ consisting of the values for a subset of attributes $\mathcal{A}' \subset \mathcal{A}$. For simplicity, we also assume that the adversary knows (1) that the target is in the dataset ($u \in U$), and (2) that the target's record is unique in the dataset, given all known attributes ($\forall v \in U, v \neq u :\ r^u_{\mathcal{A}'} \neq r^v_{\mathcal{A}'}$).

Formally, the adversary's goal is to devise both a multiset of queries $(q_1, \ldots, q_k) \in \mathcal{Q}^k$ and a rule $G : \mathbb{R}^k \to \{0, 1\}$ to combine the answers to the queries $R(D, q_1), \ldots, R(D, q_k)$ to retrieve the sensitive attribute:

$$\hat{s} = G(R(D, q_1), \ldots, R(D, q_k)) \tag{4.2}$$

The attack is considered successful if the predicted value $\hat{s}$ for the sensitive attribute matches the correct value, i.e., $\hat{s} = s$.

We focus on users that are unique in the dataset D to simplify the evaluation of the privacy gain provided by the QBS alone. An adversary would indeed be able to perfectly infer the sensitive attribute a^* of unique users if the dataset were to be released as-is, allowing us to evaluate the privacy gain provided by the QBS. Note that technically, this property is true for a larger class of users, who are so-called *value-unique* [129]: all records who share the same known attributes have the same sensitive attribute. We focus on unique users for simplicity (who are all value-unique), although the method in this work applies similarly to value-unique

users.

4.3.3 Auxiliary knowledge on the target dataset

We consider two specific attack scenarios, with different assumptions on the adversary's knowledge about the private dataset:

- **AUXILIARY**: the adversary has access to a dataset D' similar to the private dataset D (e.g., from the same distribution).
- **EXACT-BUT-ONE**: the adversary has perfect knowledge of the private dataset D, except for the target record's sensitive attribute $r^u_{a_n}$.

The latter (EXACT-BUT-ONE) is the typical assumption made to evaluate "worst-case" attacks [29, 163], evaluating how much information a very strong adversary would be able to infer. The former (AUXILIARY), on the other hand, is typically used to evaluate privacy risks in practice, i.e., as a more realistic setup [302, 264, 312].

The adversary's knowledge allows them to define a training distribution π_{train} and a validation distribution π_{val} (that can be identical), from which to sample datasets. Datasets generated from the former are used to train the rule G, while datasets from the latter are used to estimate the fitness (test accuracy) of a multiset of queries.

In the AUXILIARY scenario, the adversary generates auxiliary datasets by sampling records uniformly at random without replacement from D' and appending the target record $r^u_{\mathcal{A}'}$ with a random value for the sensitive attribute $r^u_{a_n}$, which defines π_D. In practice, the adversary divides D' in a *training* and a *validation* dataset, which define π_{train} and π_{val}.

In the EXACT-BUT-ONE scenario, auxiliary datasets are obtained by choosing a random value for the target user's sensitive attribute, which defines π_D. In this setup, the training and validation distributions are identical.

Note that in both scenarios, we randomise the target user's sensitive attribute. This breaks possible correlations with known attributes, which could be used to infer the value of the sensitive attribute even if the target user does not contribute their data. Randomisation implies that the baseline success rate of an attack without access to data is 50% when the sensitive attribute is binary.

4.4 Attack methodology

4.4.1 Overview of QuerySnout

In this section, we present QuerySnout, our method for automating the discovery of attribute inference attacks against query-based systems. We here assume, without loss of generality, that the last attribute $a^* = a_n$ is sensitive and that the remaining attributes are known auxiliary information about the target $\mathcal{A}' = (a_1, \ldots, a_{n-1})$, i.e., the adversary knows the partial record $r^u_{a_i}, i = 1, \ldots, n-1$. We further assume, for simplicity, that the sensitive attribute is binary $\mathcal{V}_{a_n} = \{0, 1\}$.

The goal of QuerySnout is to find both a multiset of m queries $q_1, \ldots, q_m \in \mathcal{Q}$ and a rule G to combine the answers to these queries so as to accurately retrieve $r^u_{a_n}$, the target record's value for the sensitive attribute a_n.

The adversary's task can be formulated as a distinguishing game between two hypotheses [52][2]:

$$\text{Hypothesis } \mathcal{H}_0 : \text{ The target record's sensitive attribute is } r^u_{a_n} = 0$$

$$\text{Hypothesis } \mathcal{H}_1 : \text{ The target record's sensitive attribute is } r^u_{a_n} = 1$$

Informally, the adversary aims to distinguish between two "worlds": a world where the target record's sensitive attribute is 0 (hypothesis $\mathcal{H}_0$) and a world where the target record's sensitive

[2]Carlini et al. [52] applied the hypothesis testing framework to membership inference attacks, while we apply it to attribute inference attacks. To our knowledge, this framework can be traced to Kairouz et al. [173] who applied it to derive adversarial bounds for differential privacy.

attribute is 1 (hypothesis $\mathcal{H}_1$). Our method simulates the two worlds by instantiating the QBS on datasets sampled from the adversary's prior knowledge, each time appending the target user's partial record to the dataset with a random value s (0 or 1) for the sensitive attribute $(r^u_{a_1}, \ldots, r^u_{a_{n-1}}, s)$. The assumptions underlying the attribute inference attack are that (1) the behavior of the QBS, as captured by answers to a set of queries $q_1, \ldots, q_m$, leads to different distributions $\mathcal{D}_s$ for $s = 0$ and $s = 1$ and (2) that these differences can be captured by an adversary. Not all query sets $q_1, \ldots, q_m$ may lead to a better than random attack, as they may have no signal, e.g., because they are all suppressed (returning an answer of 0) due to selecting too few users, a mechanism known as user set size restriction. For all the QBSs we study in this work, we empirically find that at least one set of queries exists that (1) satisfies the syntax of Eq. 4.2.1 and (2) leads to different distributions of answers and hence a better than random attribute inference attack (see Sec. 4.6 for results).

Consider a QBS that adds Gaussian noise to every answer: $R(D, q) = T(D, q) + \mathcal{N}(0, 1)$ which is then rounded to the nearest integer. From the Adult dataset [272], we randomly select five attributes and one target record which is unique for these attributes. We denote by $v_1, \ldots, v_5$ the values of this record for attributes $a_1, \ldots, a_5$. We then generate 10,000 datasets of size 8,000 from the Adult dataset, each time sampling 7,999 records uniformly without replacement and appending the target record $(v_1, \ldots, v_5, s)$ with $s = 0$ or 1 uniformly at random.

Fig. 4.1 shows the distributions of query answers computed on datasets where $s = 0$ and $s = 1$ for two counting queries q_1 and q_2 and their difference. We write, for instance, "$a_1 = v_1 \wedge a_2 != v_2 \wedge a_4 = v_4 \wedge s = 1$" to represent the query COUNT(*) WHERE $a_1 = v_1$ AND $a_2 \neq v_2$ AND $a_4 = v_4$ AND $s = 1$, counting all the users having values v_1 and v_4 for the first and fourth attributes, a value different from v_2 for the second attribute, and a sensitive attribute of 1. There are two sources of randomness responsible for the variability in the query answers: dataset sampling and noise addition by the QBS. This randomness captures the uncertainty of the adversary having imperfect knowledge of the dataset and of the noise added. As expected, answers for the first query q_1 are larger by 1 on average when $s = 1$, as q_1 selects the target record if and only if $s = 1$. However, this difference of 1 is small when compared with the standard deviation (of approximately 5.3), meaning that the distributions are hard to distinguish, and

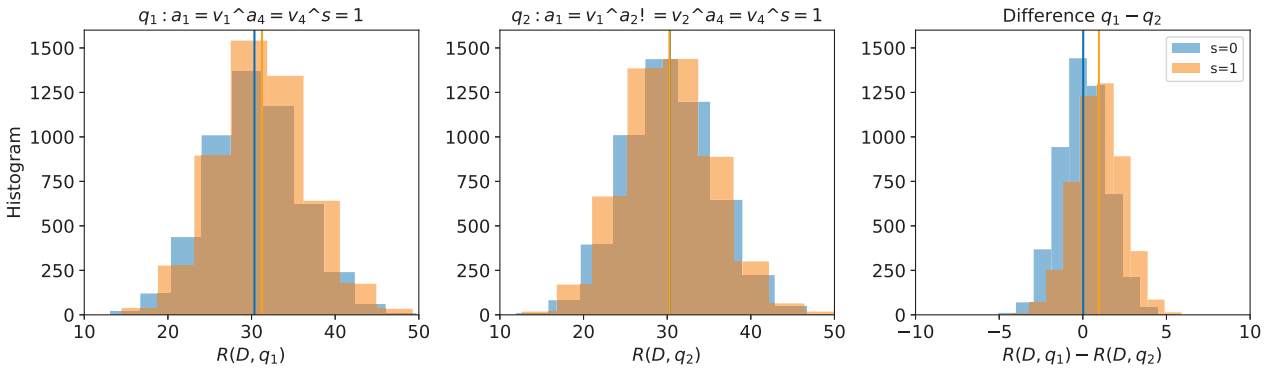


Figure 4.1: **Distribution of answers to two queries and their difference computed on datasets with $s = 0$ and $s = 1$.** In this example, the answers are retrieved from a simple QBS with Gaussian noise addition and rounding.

thus that this query alone would not lead to a successful attack. Answers to the second query q_2 are distributed the same for $s = 0$ and $s = 1$ as this query never selects the target user, because of the condition $a_2 \neq v_2$. While this may lead to considering q_2 does not have useful signal at all, this query can become useful when combined with q_1. Indeed, it turns out that the target record is also unique for attributes a_1, a_2 and a_4 in the Adult dataset, and by extension in all subsampled datasets. This means that for the same dataset, the true counts of the two queries differ by 1 when $s = 1$ and by 0 when $s = 0$. Importantly, the difference between the two queries is significantly less noisy (standard deviation of approximately 1.4) than each of the queries taken individually, because the randomness of the sampling of the datasets is cancelled, leaving only the randomness of Gaussian noise addition.

Our goal is to automatically discover a set of queries of relatively small size m whose answers are distributed significantly differently under the two hypotheses. However, evaluating exactly to what extent two distributions can be distinguished is challenging, especially in high dimensions (i.e., for many queries). We tackle this challenge by training a classifier to infer the sensitive attribute (i.e., hypothesis $\mathcal{H}_s$ with $s = 0$ or 1) based on query answers, and use its accuracy as a score capturing the extent to which the two distributions can be distinguished. This also has a natural adversarial interpretation, since the classifier can be seen as a rule used by the adversary to infer sensitive information from the answers they received. In this view, the accuracy is the likelihood that the adversary's guess is correct. We apply the classifier to the target QBS to infer the most likely value of the target record's sensitive attribute in the target dataset.

We frame the discovery of attacks against a QBS as an optimisation problem over a search space of solutions consisting of all multisets of m queries of a specific structure (Sec. 4.4.2). QuerySnout optimises solutions in this space with regards to the accuracy of the attribute inference attack obtained by applying an automatically trained rule G on the answers to this multiset of queries (Sec. 4.4.3). To optimise for attacks, QuerySnout uses an evolutionary algorithm based on a novel mutation algorithm (Sec. 4.4.4).

4.4.2 Search space of solutions

Given a restricted *query search space* $\mathcal{Q}_s \subset \mathcal{Q}$, we define a *solution* as an unordered list of m queries $q_1 \ldots, q_m \in \mathcal{Q}_s$. Formally, solutions are multisets of queries, meaning that the same queries can be repeated multiple times. We denote by $\mathcal{S}^m$ the set of all *solutions* that is explored by the evolutionary algorithm. We describe it formally as follows:

$$\begin{aligned}\mathcal{S}^m := \Big\{ &\{(q^1, m_1), \ldots, (q^k, m_k)\} : q^1, \ldots, q^k \in \mathcal{Q}_s^n, \\ &i \neq j \implies q_i \neq q_j, \forall i, m_i \in \mathbb{N}, \ \sum_{i=1}^{k} m_i = m \Big\}\end{aligned} \tag{4.3}$$

where m_i denotes the multiplicity of the i-th unique query.

Note that we here assume that the order of the queries does not impact the way the QBS answers them. All the systems we consider satisfy this property. In the discussion, we explain how our method could be adapted to systems for which query order matters.

Query search space. For general query syntaxes (e.g., SQL), the query space $\mathcal{Q}$ is extremely large, even possibly unbounded. In this work, we restrict – as a starting point – the search to a simple, yet still very large *query space* $\mathcal{Q}_s \subset \mathcal{Q}$ consisting of counting queries that select records via a conjunction of up to n simple conditions, with no more than one condition per attribute.

Given a set of condition operators $\mathcal{C}_s$ (such as "equal to" or "different from"), we write $a_i \ c_i \ v_i$ to denote a condition on the i-th attribute of operator $c_i \in \mathcal{C}_s$, with value $v_i \in \mathcal{V}_{a_i}$ belonging to set

of acceptable values for the i-th attribute. To simplify the search and exploit the information available to the adversary about the target record, we restrict conditions relating to a known attribute to use the target record's value $v_i = r^u_{a_i}, i \in \mathcal{A}'$. A condition on the unknown, sensitive attribute may use any of the acceptable values $v_n \in \mathcal{V}_{a_n}$. Formally, the query search space we consider, Q_s, consists of queries of the form:

$$\begin{aligned} &\text{SELECT COUNT(*)} \\ &\text{WHERE } a_1\ c_1\ r^u_{a_1} \text{ AND } \ldots \text{ AND } a_{n-1}\ c_{n-1}\ r^u_{n-1} \text{ AND } a_n\ c_n\ v_n \end{aligned} \tag{4.4}$$

In this work, we set the set of condition operators to $\mathcal{C}_s = \{\neq, =, \perp\}$, meaning that conditions can be of operator $=$ (equal to), $\neq$ (different from) or $\perp$ indicating that there is no condition on the i-th attribute. In other words, a condition relating to the i-th attribute can be: (1) $a_i = v_i$, (2) $a_i \neq v_i$ or (3) no condition, which we write as $a_i \perp v_i$ as a convention (the value is ignored); with $v_i \in \mathcal{V}_{a_i}$.

Finally, we assume – for simplicity – the sensitive attribute to be binary: $\mathcal{V}_{a_n} = \{0, 1\}$. Under this assumption, the conditions $a_n = 0$ and $a_n \neq 1$ are equivalent, and there is a one-to-one mapping between the query search space $\mathcal{Q}_s$ and ${\mathcal{C}_s}^n$, as one choice of condition operators $(c_1, \ldots, c_n) \in \mathcal{C}_s^n$ corresponds to exactly one query with operator (2) and conditions $a_1\ c_1\ r^u_{a_1},\ a_2\ c_2\ r^u_{a_2},\ \ldots,\ a_{n-1}\ c_{n-1}\ r^u_{a_{n-1}}$ and $a_n\ c_n\ 0$, respectively. The search algorithm can thus represent queries as strings of k operators.

As an example, let attributes a_1, a_2 and a_3 denote a person's *age*, *nationality* and *diagnosis* for a disease. The query with condition operators $(c_1, c_2, c_3) = (=, \perp, \neq)$ applied to a target record of known age and nationality $(v_1, v_2, v_3) = (19, \mathit{Romania}, 0)$ yields the query: "How many people in the database have an age of 19 and a positive diagnosis?" Note that there is no condition on the *nationality* attribute and that the $\neq$ operator on the sensitive attribute is equivalent to selecting users with a positive diagnosis.

We restrict the query space for two main reasons. First, a large number of attacks from previous work can be performed using only queries expressed this way, and we can thus compare our results with manual attacks. For instance, averaging attacks [96], difference attacks [97] and

the differential noise-exploitation attack against Diffix [129] (described in Sec. 4.5.4) can all be written with queries in $\mathcal{Q}_s$. Second, the problem is already computationally challenging, as the size of the attack search space is very large. Indeed, when the sensitive attribute is binary, the cardinality of the query space is $|\mathcal{Q}_s| = |\mathcal{C}^n| = 3^n$, which increases exponentially with the number of known attributes n. The size of the attack search space $\mathcal{S}^m$ is therefore equal to the number of multisubsets with m elements[3] from a set of size 3^n [119]: $\binom{3^n+m-1}{m} \approx \frac{3^{n(m-1)}}{m!}$, where the approximation holds if $m \ll 3^n$. For typical values such as those used in this work, e.g., $m = 100$ and $n = 6$, an exact computation yields an extremely large attack search space size of $\approx 1.33 \times 10^{131}$. This emphasises the importance of being able to search for solutions efficiently.

4.4.3 Automated learning of a rule G

In order to evaluate whether a solution can be used to perform an attack, we propose a method to automatically learn a rule G to combine the answers to the queries of a solution $\{q_1, \ldots, q_m\} \in \mathcal{S}^m$ to perform an attribute inference attack (Fig. 4.2). Our method uses the training distribution π_{train} to sample d auxiliary datasets $D'_1, \ldots, D'_d \sim \pi_{\text{train}}$. Each dataset is protected by a QBS initialised with different seeds for their random number generator. We perform the queries on each QBS to obtain the answers $(R(D'_i, q_1), \ldots, R(D'_i, q_m))$, $i = 1, \ldots, d$. We denote by X_{train} the dataset of d samples of m features each (a feature is a query answer) obtained in this way. We also denote by Y_{train} the d values for the target record's sensitive attribute in the corresponding auxiliary datasets. Finally, we train a binary classification model G on $(X_{\text{train}}, y_{\text{train}})$ to infer the value of the sensitive attribute from the answer to queries. The model will be used as the rule G to combine the answers to queries in the solution.

Our approach assumes that an optimal rule G can be effectively approximated using machine learning models. For instance, the averaging attack described in Sec. 4.5.4 computes a linear combination of the query answers, which can be represented by a logistic regression. Furthermore, multilayer perceptrons are known to be universal approximators [156], which makes them even more apt to model an arbitrary rule. Our approach to learn the rule G automatically

[3]Note that the size of $\mathcal{S}^m$ is *not* $|\mathcal{Q}_s|^m$, as the order does not matter.

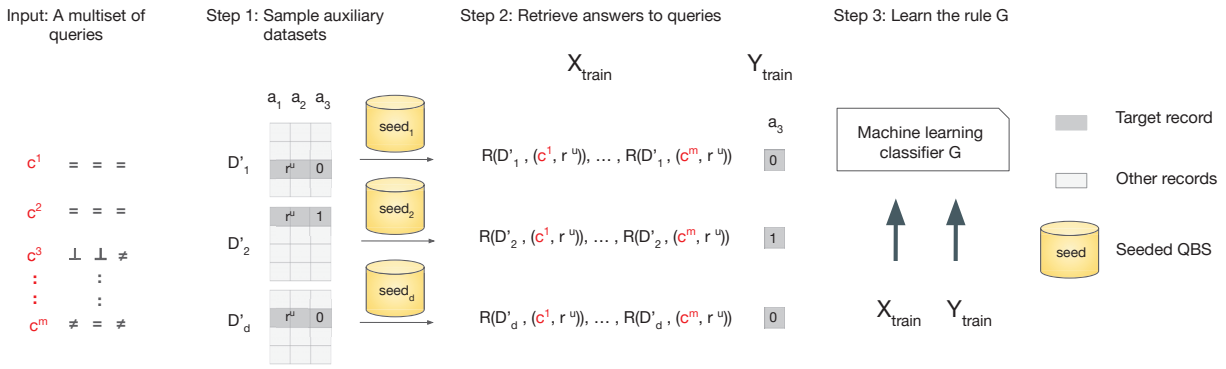


Figure 4.2: **Automated learning of a rule G.** We illustrate our approach to automatically combine the answers to a multiset of queries, via a rule G learned using a machine learning model. In our example, partial records r^u consist of values for attributes a_1 and a_2, while a_3 is the sensitive attribute. The adversary samples d auxiliary datasets (randomising the value of a_3 for the target record), then retrieves $d \times m$ answers to the queries (in red) from (differently seeded) QBSs protecting the datasets. The answers together with the target's sensitive attributes from the corresponding datasets are used to train a classifier G.

given a set of queries is similar to the one proposed by Pyrgelis et al. to perform membership inference attacks on aggregate statistics [264], except we apply it to attribute inference attacks and use a broader and more flexible range of queries.

4.4.4 Search space exploration

We explore the search space for queries that can be combined to perform highly accurate attacks using evolutionary algorithms. Our approach (illustrated in Fig. 4.3) improves a *population* of solutions over time by applying small random changes, called *mutations*, to the solutions. We propose a mutation operator tailored to the task that allows our procedure to efficiently explore the search space. The solutions are optimised with regards to their *fitness*: their ability to infer the target record's sensitive attribute.

Algorithm 2 details our procedure to explore the space of attacks using evolutionary algorithms. We start from a population of P solutions $s_1, \ldots, s_P$. Each solution consists of m queries $s_j = (c^{j1}, \ldots, c^{jm})$, which are initialised uniformly at random: $c^{ji} \leftarrow \mathcal{U}(\mathcal{C}^n), i = 1, \ldots, m$. The algorithm runs for N generations. In each generation, we first evaluate the fitness of each of the P solutions. The fitness function, which we describe in the next paragraph, estimates the

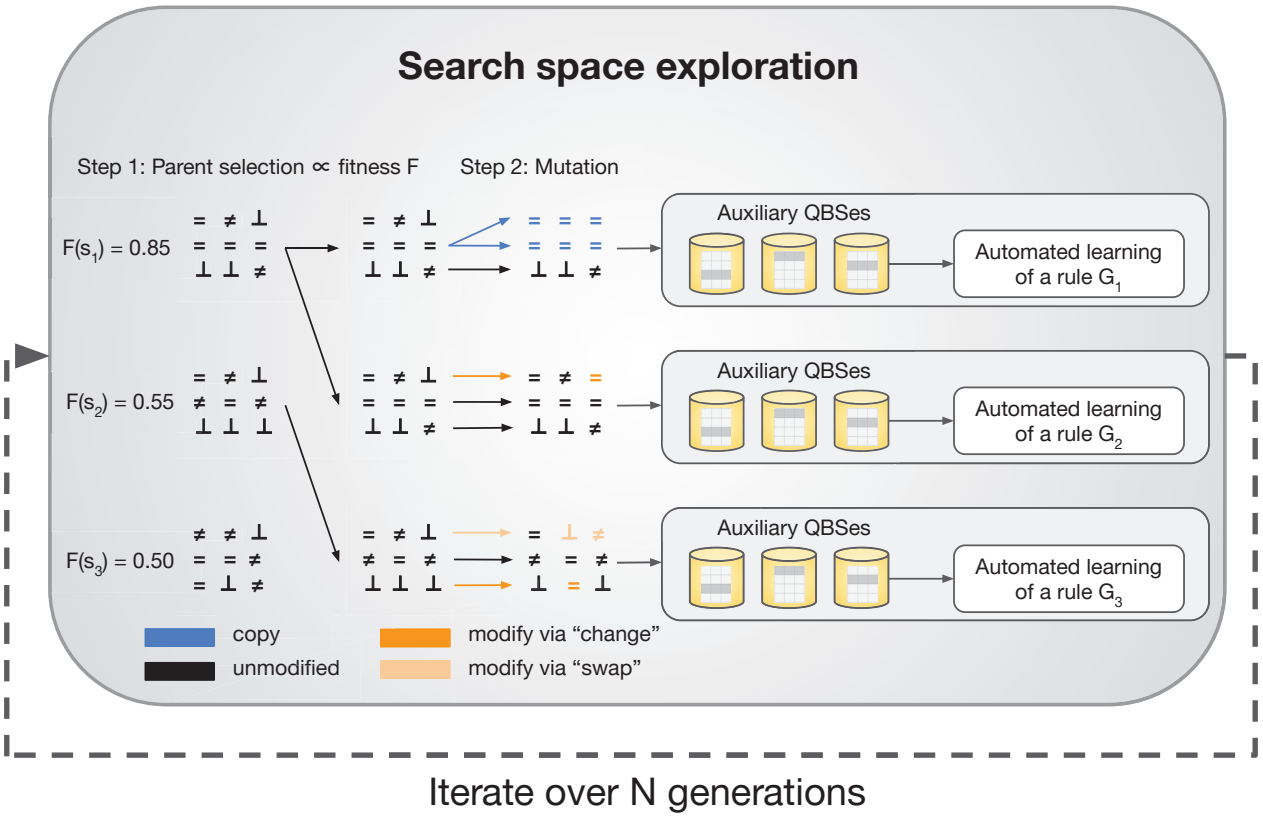


Figure 4.3: **Illustration of how QuerySnout explores the search space using evolutionary algorithms.** In this example, the search space consists of multisets of $m = 3$ queries relating to $n = 3$ attributes. Solutions with higher fitness are more likely to be selected. We use different colours to illustrate the changes to a query and black for unmodified query operators.

accuracy of the attribute inference attack obtained with the queries in the solution. Second, we sort the population decreasingly according to the fitness of each solution. Third, we create a new population consisting of (1) a number P_e of *elites* [88], i.e., the P_e solutions in the current population with the highest fitness, and (2) a number $P - P_e$ of *offsprings*. Each offspring is generated by applying a mutation to a *parent* solution.

The parents are selected from the current population by sampling with replacement using the *biased roulette wheel* [288]: the probability to sample a solution is equal to its fitness divided by the sum of fitnesses of solutions in the population. This ensures that solutions with higher fitness are more likely to generate offsprings.

Fitness evaluation. Algorithm 3 describes our procedure to evaluate the fitness. The procedure uses training and validation auxiliary datasets $D'_{\text{aux}} = (D'_{\text{train}}, D'_{\text{val}})$. The datasets in D'_{train}

Algorithm 2 QUERYSNOUTEVOLUTIONARYSEARCH

Inputs:
P: Population size (number of solutions).
m: Number of queries in a solution.
P_e: Number of elites.
N: Number of generations.
D'_{aux}: Auxiliary datasets used to evaluate the fitness.
p_{mut}: Mutation parameters.

Output:
population: A population of P solutions in the search space of solutions $\mathcal{S}^m$.

Initialise:
$population \leftarrow [random\_solution(m)$ for $i = 1$ to $P]$
for $g = 1$ to N **do**
  $fitnesses \leftarrow [\text{EVALUATEFITNESS}(solution, D'_{\text{aux}})$ for $solution$ in $population]$
  `// Sort the population by fitness.`
  $sort\_descending(population, fitnesses)$
  `// Pass the elites unchanged to the next generation.`
  $new\_population \leftarrow population[: P_e]$
  for $i = 1$ to $P - P_e$ **do**
    $parent \leftarrow select\_parent(population, fitnesses)$
    $offspring \leftarrow \text{APPLYMUTATION}(parent, p_{\text{mut}})$
    $new\_population.append(offspring)$
  end for
  $population \leftarrow new\_population$
end for

and D'_{val} are (1) sampled upon initialisation from π_{train} and π_{val}, respectively, then (2) protected by individual QBS instances with different seeds initializing their random number generator. To evaluate the fitness, we train a rule G to combine the answer of queries to predict the sensitive attribute of the target record, using auxiliary datasets D'_{train}, as described in Sec. 4.4.3. We measure the accuracy of this prediction on training and validation data, a_{train} and a_{val} respectively. We use $\min(a_{\text{train}}, a_{\text{val}})$ as fitness in order to minimise the effect of randomness in the evaluation. Indeed, the accuracy estimates are noisy, and we found empirically that using a_{val} as fitness leads to the algorithm selecting solutions that were *lucky* when estimating the fitness, rather than really superior. The effect was however minor, occurring mostly when the algorithm had converged.

Mutation operator. The mutation operator aims to explore the local space around known good solutions. Algorithm 4 presents the pseudocode. The mutation operator takes as input

Algorithm 3 EvaluateFitness

Inputs:
solution: A solution in the search space $\mathcal{S}^m$. $D'_{\text{aux}} = (D'_{\text{train}}, D'_{\text{val}})$: Two lists of auxiliary datasets used to train and evaluate the fitness function.
Output:
$fitness$: Accuracy of an ML model trained to predict the sensitive attribute of a record.
Initialise:
`// Answers to queries used as input features`
`// for training and evaluation.`
$X_{\text{train}}, X_{\text{val}} \leftarrow empty\_list, empty\_list$
`// Corresponding sensitive attributes.`
$y_{\text{train}}, y_{\text{val}} \leftarrow empty\_list, empty\_list$
for split $\in \{\text{train}, \text{val}\}$ **do**
  for $k = 1$ to $length(D'_{\text{split}})$ **do**
    $answers \leftarrow empty\_list$
    for c in *solution* **do**
      $q = \text{COUNT WHERE} \bigwedge_{i=1}^{n-1} (a_i\, c_i\, r^u_{a_i}) \wedge (a_n\, c_n\, 0)$
      $answers.append(R(D'_{\text{split}}[k], q))$
    end for
    $X_{\text{split}}.append(answers)$
    $y_{\text{split}}.append(r^u_{a_n})$ `// Extracted from` $D_{\texttt{split}}[k]$.
  end for
end for
$G = train\_model(X_{\text{train}}, y_{\text{train}})$ `// Train the rule` G.
`// Evaluate the model on each split.`
$\hat{y}_{\text{train}}, \hat{y}_{\text{val}} \leftarrow G.predict(X_{\text{train}}), G.predict(X_{\text{val}})$
$fitness\_train, fitness\_val \leftarrow acc(\hat{y}_{\text{train}}, y_{\text{train}}), acc(\hat{y}_{\text{val}}, y_{\text{val}})$
$fitness\_val \leftarrow accuracy(\hat{y}_{\text{val}}, y_{\text{val}})$
$fitness \leftarrow min(fitness\_train, fitness\_val)$

a *parent* solution and makes small changes to its queries in order to generate an *offspring* solution. Each query in the parent solution is treated separately, and is either copied with probability p_{copy}, modified in place with probability p_{modify}, or left unmodified with probability $1 - p_{\text{copy}} - p_{\text{modify}}$. Note that copying a query means that we first add the query – as is – to the offspring and then perform the following on a copy of it. If the system is deterministic, we modify the copy; otherwise, with equal probability we either keep the unmodified copy or replace it with a modified version. We add the resulting query to the offspring. We distinguish between the two cases because asking a query repeatedly provides no additional information when the QBS is deterministic. Finally, as the offspring may now contain more than m queries (due to copying), we select a random subset of m queries. This mutation operator is inspired

Algorithm 4 ApplyMutation: Mutation algorithm.

Inputs:
$parent$: A solution in the attack search space $\mathcal{S}^m$.
p_{copy}: Probability to copy a query.
p_{modify}: Probability to modify a query.
p_{change}: Probability to change an operator in a query.
p_{swap}: Probability to swap two operators in a query.
Output:
$offspring$: New solution in the attack search space $\mathcal{S}^m$.
Initialise:
$offspring \leftarrow empty\_list$
for $i = 1$ to m **do**
  $c \leftarrow solution[i]$
  $u \leftarrow \mathcal{U}(0, 1)$
  `// Case 1: Copy the query.`
  if $0 \leq u < p_{\text{copy}}$ **then**
    $offspring.append(c)$
    `// Case 1a: Modify the copy.`
    if $qbs\_deterministic$ **then**
      $c' \leftarrow \text{ModifyQuery}(c, p_{\text{change}}, p_{\text{swap}})$
      $offspring.append(c')$
    else
      `// Case 1b: Modify the copy or add the query unchanged with equal chance.`
      $u' \leftarrow \mathcal{U}(0, 1)$
      if $u' > 0.5$ **then**
        $c' \leftarrow \text{ModifyQueryRule}(c, p_{\text{change}}, p_{\text{swap}})$
        $offspring.append(c')$
      else
        $offspring.append(c)$
      end if
    end if
  end if
  `// Case 2: Modify the query.`
  if $p_{\text{copy}} \leq u < p_{\text{copy}} + p_{\text{modify}}$ **then**
    $c' \leftarrow \text{ModifyQuery}(c, p_{\text{change}}, p_{\text{swap}})$
    $offspring.append(c')$
  end if
  `// Case 3: Add the query unchanged.`
  if $p_{\text{copy}} + p_{\text{modify}} \leq u$ **then**
    $offspring.append(c)$
  end if
end for
`// Keep a random subset of` m `queries.`
$offspring \leftarrow random\_permutation(offspring)[:m]$

Algorithm 5 ModifyQuery: Algorithm to modify a query.

Inputs:
$c = (c_1, \ldots, c_n)$: Query to be modified.
p_{change}: Probability to switch an operator.
p_{swap}: Probability to swap two query operators.
Output:
new_c: A modified version of the query c.
Initialise:
$new\_c \leftarrow (c_1, \ldots, c_n)$
// Randomise the attribute order.
$\sigma \leftarrow random\_permutation(n)$
$swapped \leftarrow empty\_set$
for $i = 1$ to n **do**
  // Skip already swapped attributes.
  if $swapped.contains(i)$ **then**
    $continue$
  end if
  $u \leftarrow \mathcal{U}(0, 1)$
  // Case 1: Change the operator.
  if $0 \leq u < p_{\text{change}}$ **then**
    $new\_c[\sigma(i)] \leftarrow \mathcal{U}(\mathcal{C}_s \setminus \{c_i\})$
  end if
  // Case 2: Swap two operators.
  if $p_{\text{change}} \leq u < p_{\text{change}} + p_{\text{swap}}$ **then**
    $new\_c[\sigma(i)] \leftarrow c_{\sigma(i+1)}$
    $new\_c[\sigma(i+1)] \leftarrow c_{\sigma(i)}$
    $swapped.add(i+1)$
  end if
  // Case 3: Leave the query unchanged.
end for

by the fact that many attacks on QBSs [96, 129] use pairs of similar queries, or repeat the same queries multiple times.

Modifying a query. We now describe our procedure to modify a query consisting of n operators $c = (c_1, \ldots, c_n) \in \mathcal{C}^n$ (and refer the reader to Algorithm 5 for the pseudocode). For each attribute a_i, we either (1) "change" the corresponding operator c_i with probability p_{change} by replacing it with a different value in the operator set which we sample uniformly at random, (2) swap the operators between the i-th attribute and another attribute that has not been swapped yet with probability p_{swap}, or (3) leave the operator unchanged. To ensure that no attribute pair is more likely to be swapped compared to the others, we randomly permute the order in which this procedure considers the entries of c. We introduce the "swap" operation to

exploit symmetry between attributes: if a given attack is successful, it is likely that swapping the conditions of two attributes will lead to a similarly successful attack. Note that although "swap" and "change" can produce the same outcome, they do so with different probabilities and are thus not equivalent.

Choosing the mutation parameters. The mutation parameters are set such that, on average, we copy (modify) a fraction $p_{\text{copy}}(p_{\text{modify}})$ of the m queries in a solution, a common heuristic in evolutionary search. The same heuristic also applies to parameters p_{swap} and p_{change}. In practice, the ranges of mutation parameters should be informed by the exploration-exploitation trade-off: making more changes to a solution allows to explore the space more, but making too many changes at once might hinder the improvement of solutions over time.

4.5 Experimental setup

In this section, we first present the datasets we use to evaluate our automated attacks. Second, we present how we instantiate the auxiliary knowledge in the AUXILIARY and EXACT-BUT-ONE scenarios. Third, we describe the query-based systems against which we evaluate our automated attacks. Finally, we describe the system-specific manual attacks we compare against.

4.5.1 Datasets

We use three publicly available datasets: Adult [272], Census [273] and Insurance [274]. Adult contains 48482 individual records of 14 socio-demographic attributes each. Census contains 299285 records of 41 socio-demographic attributes each. Demographic attributes are, for instance, "age", "education" and "occupation". We assign "income" as the binary sensitive attribute for both Adult and Census and randomise it (50:50) as described below. Insurance contains 9822 records of 86 attributes about customers of a car insurance company, aggregated at the zipcode level. We only use the 43 socio-demographic attributes and add a randomised (50:50) sensitive attribute.

4.5.2 Instantiating the auxiliary knowledge

For each dataset, in each repetition of the experiments, we sample $n-1 = 5$ attributes uniformly at random without replacement and discard the others. These are the attributes $\mathcal{A}'$ whose values for a target record we assume to be known to the adversary. We then randomly partition the dataset between a training auxiliary dataset D_{train}, a validation auxiliary dataset D_{val}, and a testing dataset D_{test} of equal sizes. We select 100 unique *target records* from D_{test} uniformly at random without replacement. For each target record, we instantiate the training and validation distributions π_{train} and π_{val}. We also instantiate a test distribution π_{test} from which *private* datasets D containing the target record will be sampled in two steps.

- In the AUXILIARY scenario, for split $\in \{\text{train}, \text{val}, \text{test}\}$, we instantiate π_{split} on the corresponding dataset D_{split} as described in Sec. 4.3.3. We also add a randomised sensitive attribute (50:50) to datasets sampled from π_{split}, and remove duplicates of the target record from all datasets generated to ensure that the it is unique.
- In the EXACT-BUT-ONE scenario, we sample one dataset D from D_{test} and add a randomised sensitive attribute (50:50). We then instantiate π_{train}, π_{val} and π_{test} on D as described in Sec. 4.3.3.

To ensure that information from the test distribution does not unintendedly leak into the train and validation distributions, the seeds used to instantiate the QBSs on datasets sampled from $\pi_{\text{train}}, \pi_{\text{val}}$ and π_{test} are *all distinct*. To evaluate the actual privacy leakage of the query-based system and in line with previous work [129], we choose to break the correlations between the sensitive attribute and the other attributes by randomizing the binary sensitive attribute with a 50:50 distribution.

4.5.3 Query-based systems

We describe our implementation of three query-based systems. The first two are based on real-world privacy protection mechanisms used by Diffix [123] and TableBuilder [125], while

the third one is a non-deterministic system combining two QBS building blocks: query set size restriction and (Gaussian) noise addition.

These systems all rely on *query set size restriction* (QSSR), where the QBS refuses to answer a query if it concerns less than T users, for some threshold T. QSSR aims at preventing queries that reveal information about a small number of users, such as counting the number of record identical to the target user and either value of s. We here assume that a QBS doing bucket suppression returns 0 instead of an error message, as this makes the attack harder by giving less information to the adversary. For simplicity, we introduce the following notation for bucket suppression: $\beta_\tau(D, Q) = I\{T(D, Q) > \tau\}$, where I is the indicator function. We also introduce the notion of *query set* $U(D, Q)$, the set of all users who satisfy the conditions of the query Q in the dataset D. Note that for counting queries, $T(D, Q) = |U(D, Q)|$.

Rounding. For realism, since we focus on counting queries, we further assume that the answers obtained from all mechanisms are then rounded to the nearest integer, and that if the mechanism would return a negative answer it returns 0 instead. This again makes the attack harder by giving less information to the adversary. When applying our attack on a mechanism, we thus use R_{qbs} defined as:

$$R_{\text{qbs}}(Q, D) = \lfloor \max(R_{\text{mechanism}}(Q, D), 0) \rceil \tag{4.5}$$

Randomness. Both Diffix and TableBuilder rely on *seeded* noise, i.e., noise produced by a pseudo-random number generator (PRNG) which outputs the same noise given the same input seed. Hence, these two systems are deterministic: for the same input query, they always output the same result. On the contrary, SimpleQBS is non-deterministic, and sample fresh random variables for each query.

Diffix. Diffix is a commercial QBS developed by the startup Aircloak [123]. It uses bucket suppression with a noisy threshold and two layers of so-called *static* and *dynamic* noise addition. All noises are seeded with different elements of the query, in order to prevent specific attacks.

Specifically, we attack Diffix-Birch [124], which adds noise to counting queries as:

$$R_{\text{Diffix}}(D,Q) = \beta_2(D,Q)\beta\tau(D,Q)\left(T(D,Q) + \sum_{i=1}^{n_{\text{cond}}(Q)} N_i^S + N_i^D\right) \tag{4.6}$$

where $n_{\text{cond}}(Q)$ is the number of non-empty conditions in Q (i.e., $\sum_{i=1}^{n} I\{o_i \neq \perp\}$), $\tau \sim \mathcal{N}(4, 0.5)$ is seeded with the query set, $N_i^S \sim \mathcal{N}(0,1)$ is seeded with the text of condition i of Q, and $N_i^D \sim \mathcal{N}(0,1)$ is seeded with the text of condition i of Q and the query set $U(D,Q)$.

TableBuilder. TableBuilder is a QBS developed by the Australian Bureau of Statistics for census contingency tables [125]. We here focus on the privacy mechanism used on individual cells (similarly to [20]), and hence attack a simplified subset of the syntax of the real system. We define this mechanism as

$$R_{\text{Tablebuilder}}(D,Q) = \beta_4(D,Q) \cdot (T(D,Q) + U), \tag{4.7}$$

where $U \sim \mathcal{U}\{-2,\ldots,2\}$ is seeded with the query set $U(D,Q)$.

SimpleQBS. We study a simple QBS that combines two common building blocks: Gaussian noise addition (with variance σ) and query set size restriction. Formally, the mechanism SimpleQBS(τ,σ^2) answers queries as:

$$R_{\text{Simple}}(D,Q) = \beta_\tau(D,Q) \cdot (T(D,Q) + N)\,, \tag{4.8}$$

with $N \sim \mathcal{N}(0,\sigma^2)$. A fresh noise sample is drawn every time a query is performed, even if the query is repeated. We instantiate this QBS with all pairs (τ,σ), $\tau = 0,\ldots,4$ and $\sigma = 0,\ldots,4$.

Note that SimpleQBS$(\tau,\sigma>0)$ can be seen as an application of the Gaussian mechanism with sensitivity $\max(1,\tau)$ [108], and thus provides $(\varepsilon,\delta)-$differential privacy for some values of ε,δ. However, for our choice of parameters, and since we do not restrict the number of queries, the corresponding values of ε,δ are very large and do not represent meaningful privacy guarantees. In Sec. 4.7, we propose an extension of our method to handle budget-based mechanisms.

4.5.4 Manual attribute inference attacks

We here describe manual attribute inference attacks from prior work against Diffix, TableBuilder, and SimpleQBS. We compare QuerySnout with these attacks both quantitatively (i.e., the accuracy of the inference) and qualitatively (i.e., what is the attack exploiting). We first introduce the notion of *difference attack*, which underlies most of the attacks we consider here.

Difference attacks. A common class of attacks against QBSs is *difference attacks*, which consist of a pair of queries (q_1, q_2) of the form

$$q_1 = \text{COUNT WHERE} \bigwedge_{i \in A'} (a_i = r^u_{a_i}) \wedge a_n = s \tag{4.9}$$

$$q_2 = q_1 \wedge a_{i'} \neq r^u_{a_{i'}} \tag{4.10}$$

where $A' \subset \{1, \ldots, n-1\}$ is a subset of attributes, $i' \notin A', i' \neq n$ is another attribute, and $s \in \{0, 1\}$ is a possible value for the sensitive attribute. The insight behind such attacks is that if the target record is uniquely identified by attributes $A' \cup \{i'\}$, then the true counts of q_1 and q_2 differ by 1 if and only if $r^u_{a_n} = s$, and 0 otherwise. The rule to combine the answers to q_1 and q_2, as well as how to select the parameters (A', i', s) depends on the system.

Diffix. Gadotti et al. [129] proposed the only known attribute inference attack against Diffix. This attack uses difference queries as described above, and exploits the structure of Diffix's noise to combine results. Specifically, if the target record is uniquely identified by attributes $A' \cup \{i'\}$, then the distribution of $R(q_1, D) - R(q_2, D)$ is either $\mathcal{N}(0, 2)$ if $r^u_{a_n} = 1$ or $\mathcal{N}(1, 2|A'|+2)$ otherwise. The attack uses a likelihood ratio test to distinguish between the cases. In order to find values of (A', i', s) such that the target record is unique and the queries are not suppressed, the attack performs a search over subsets of attributes using access to the target QBS (protecting the private dataset D) and heuristics to determine whether the assumptions are verified. We implement their attack with the `ValueUnique` [129] heuristic in the AUXILIARY setup, and the exact uniqueness oracle in the EXACT-BUT-ONE setup to reflect the additional knowledge of the adversary. Note that this attack is *interactive*, as it uses iterative access to the target QBS to choose which queries to perform.

TableBuilder. Chipperfield et al. [67] propose a simple difference attack against TableBuilder. For each known attribute j and value $s \in \{0, 1\}$, they perform the difference queries $(q_1^{j,s}, q_2^{j,s})$ for $A' = \{1, \ldots, n-1\} \setminus \{j\}$ and $i' = j$. They then compute the difference $r^{j,s} = R(q_1^{j,s}, D) - R(q_2^{j,s}, D)$, and observe that if $r^{j,s} \geq 5$, the true difference must be 1 (because the noise is bounded), and the attack predicts that the user's target value is s. If $r^{j,s} < 5$, $\forall j, s$, the attacks computes the averages $\mu_s = \frac{1}{n-1} \sum_{j=1}^{n-1} r_{j,s}$ and predicts 0 iff $\mu_0 > \mu_1$ (because the noise is centred).

Rinott et al. [277] propose another attack, based on the same queries but using a different combination rule. This second attack exploits the fact the noise added to a query only depends on the userset, and the same noise is thus added to all the queries selecting the same userset. Then, if the answer to both queries in a pair are equal (i.e., $r^{j,s} = 0$), the attack predicts $1 - s$ (since the target user is not in the user set of either query, $r^u_{a_n} \neq s$). Our work is, to the best of our knowledge, the first to empirically evaluate this attack, since the original paper only presented it as a theoretical vulnerability. We implement both attacks with a small modification to only take into account non-suppressed queries. Additionally, we adapt the attacks to the EXACT-BUT-ONE scenario to reflect the additional knowledge of the adversary, by selecting attribute subsets for which the target record is unique (since they have perfect knowledge of D).

SimpleQBS. Non-deterministic noise addition and simple bucket suppression have long been known to be vulnerable to respectively averaging and difference attacks [97, 96].

When $\tau = 0$, SimpleQBS is vulnerable to a simple averaging attack, where the query $q_{\text{direct}} = \bigwedge_{i=1}^{n-1}(a_i = r^u_{a_i}) \wedge a_n = 0$ is repeated m times to obtain the results $(r_1, \ldots, r_m)$. Since the noise is centred, $\mathbb{E}[R_i] = 1$ (resp. 0) if and only if $r^u_{a_n} = 0$ (resp. 1). The adversary predicts 0 iff $\frac{1}{m} \sum_{i=1}^{m} r_i < \frac{1}{2}$, and 1 otherwise.

When $\tau > 0$, the query q_{direct} is suppressed by bucket suppression. We thus combine a difference and an averaging attack. We use the adversary's auxiliary information to find values of (A', i', s) such that the user is unique for (A', i') and the queries bypass bucket suppression in two steps. First, we generate auxiliary datasets (from the auxiliary knowledge available in each scenario).

Second, we select the pairs (q_1, q_2) for which both assumptions are satisfied for the largest fraction of datasets. When $\sigma > 0$, we repeat the best pair of queries $m/2$ times, averaging the results to obtain μ_{q_1} and μ_{q_2}. We then output s iff $\mu_{q_1} - \mu_{q_2} > \frac{1}{2}$, and $1 - s$ otherwise. When $\sigma = 0$, we select the $m/2$ best pairs and perform them in decreasing order until the result (r_1, r_2) is such that $r_1 > 0$, $r_2 > 0$ and $r_1 - r_2 \in \{0, 1\}$. We output s iff $r_1 = r_2 + 1$, and $1 - s$ otherwise.

4.6 Empirical results

The goal of our empirical evaluation is fourfold. First, we want to show that it is possible to automate the discovery of attribute inference attacks against a QBS by "pressing a button". Second, we want to understand quantitatively how well the attacks discovered by QuerySnout perform when compared to manual attacks. Third, we want to understand qualitatively what vulnerabilities are exploited by QuerySnout, for instance if they are similar to known attacks. Finally, we want to showcase the versatility of QuerySnout by deploying it on a variety of attack scenarios to derive new insights.

4.6.1 Attack parameters

For each target record, we run the evolutionary search and extract after N generations the solution s^* having the highest fitness. For computational efficiency and as the attacks against different records are independent, we parallelise the attack over the target records. We use 2000 training datasets (sampled from π_{train}), 1000 validation datasets (sampled from π_{val}), and 500 test datasets (sampled from π_{test}). The datasets are of size 8000 for Adults and Census and 1000 for Insurance.

Evolutionary search parameters. Each evolutionary search uses solutions of $m = 100$ queries, a population size of $P = 100$ and a maximum number of $N = 200$ generations with a stopping criteria of having 10 generations of fitness superior to 99.99%. To evaluate the fitness

of a solution, we use a logistic regression as the binary classifier G. We also experimented with a multilayer perceptron and found that it did not improve the performance, while significantly increasing the training time. We set the mutation parameters for modifying operators in a query to $p_{\text{change}} = \frac{1}{n} = \frac{1}{6}$, so that on average we change one operator. Similarly, we use $p_{\text{swap}} = \frac{1}{6}$, $p_{\text{copy}} = 0.025$ and $p_{\text{modify}} = 0.025$.

Attack success metric. For each target record, we compute the accuracy of the attribute inference attack defined by the best solution s^* and the trained classifier G on 500 datasets sampled from the corresponding test distribution $D \sim \pi_{\text{test}}$. Our metric for the attack success is the average accuracy over the 100 target users, which we report averaged over 5 repetitions. Note that we randomise the private attribute a_n, so that the random guess baseline has an accuracy of 50%.

4.6.2 Real-world systems

Our results show that the attacks discovered automatically by QuerySnout match and often outperform by a large margin the manual attacks from previous work.

AUXILIARY scenario. Table 4.1a and Table 4.2a show how, across all datasets and systems, QuerySnout matches or outperforms manual attacks. More specifically, it matches the accuracy of manual attack by Gadotti et al. [129] against Diffix on the Adult and Census datasets and strongly outperforms it on the Insurance dataset. Similarly, QuerySnout strongly outperforms Chipperfield et al. [67] on all datasets and matches Rinott et al. [277] on Adult and Census, while strongly outperforming it on Insurance. **Our results show how automated attacks, and QuerySnout in particular, can today not only replicate but also outperform existing manual attacks.** We identify two reasons for the good performances of QuerySnout.

First, a manual inspection of queries found by QuerySnout on Diffix shows that difference queries account for $\geq 97.0\%$ of the accuracy, while constituting less than half of the queries in solutions (see Sec 4.6.8 for a detailed analysis). Similarly, the difference queries found by QuerySnout on TableBuilder also account for $\geq 97.5\%$ of the accuracy. QuerySnout is able to

Table 4.1: **Comparison between automated and manual attacks against Diffix** on different scenarios and datasets. We report the attack accuracy over 100 target records, averaged over 5 repetitions with standard deviation.

(a) AUXILIARY	**Adult**	**Census**	**Insurance**
QuerySnout (automated)	**77.8** (0.5)	**78.3** (1.4)	**80.1** (0.6)
Gadotti et al. [129] (manual)	76.3 (0.8)	76.9 (1.4)	73.0 (1.2)
(b) EXACT-BUT-ONE	**Adult**	**Census**	**Insurance**
QuerySnout (automated)	**90.2** (0.6)	**88.3** (0.9)	**91.6** (1.2)
Gadotti et al. [129] (manual)	77.1 (0.9)	77.5 (2.0)	74.4 (0.7)

Table 4.2: **Comparison between automated and manual attacks against TableBuilder** on different scenarios and datasets. We report the attack accuracy computed over 100 target records, averaged over 5 repetitions with standard deviation.

(a) AUXILIARY	**Adult**	**Census**	**Insurance**
QuerySnout (automated)	**84.5** (0.6)	**85.5** (1.4)	**85.4** (0.6)
Rinott et al. [277] (manual)	76.1 (7.5)	78.1 (7.0)	56.9 (4.6)
Chipperfield et al. [67] (manual)	61.2 (3.5)	62.4 (3.1)	52.8 (1.8)
(b) EXACT-BUT-ONE	**Adult**	**Census**	**Insurance**
QuerySnout (automated)	**98.1** (0.7)	**96.6** (0.9)	**98.8** (0.7)
Rinott et al. [277] (manual)	83.1 (8.7)	72.1 (13.4)	76.5 (2.3)
Chipperfield et al. [67] (manual)	72.3 (6.4)	64.4 (7.2)	67.2 (1.4)

outperform the manual attacks because it is able to (1) find more attribute subsets for which the target record is likely unique, and (2) combine the results from all subsets taken together for the attack.

Second, the gap between QuerySnout and manual attacks is particularly strong on the Insurance dataset, for both QBSs. This is due to previous attacks performing worse on Insurance than on Adult and Census and to QuerySnout performing better. We hypothesise that this difference is due to the smaller size of the Insurance dataset ($|D| = 1000$ for Insurance and $|D| = 8000$ for Adult and Census), making queries more likely to be suppressed by the query set size restriction (QSSR). While our automated attacks also exploit difference attacks, we believe its flexibility allows it to find more specific queries that bypass QSSR. In Sec. 4.6.7, we show that QuerySnout performs slightly better in general on smaller datasets.

EXACT-BUT-ONE scenario. Table 4.1a and Table 4.2a show how the accuracy of the

best attack found by QuerySnout increases by at least 10% when moving from the AUXILIARY to the EXACT-BUT-ONE scenarios across datasets and QBSs. More specifically, the accuracy of QuerySnout increases by 12.4% (resp. 10.1% and 11.5%) for Adults (resp. Census and Insurance) compared to AUXILIARY against Diffix and by 13.7% (resp. 11.2% and 13.4%) against TableBuilder. It also strongly outperforms the baselines across all datasets and QBSs, with the exception of Rinott et al. on Census where it matches them. **These results show that much stronger attacks than previously believed can be devised against both Diffix and TableBuilder by a strong ("worst-case") adversary. They also show how QuerySnout is able to automatically discover new and better attacks across scenarios and datasets.**

For instance, in the case of Diffix, an adversary could perform the query $a_1 = r^u_{a_1} \wedge a_n = 0$. Denote by C the true count of this query when $r^u_{a_n} \neq 0$. The answer returned by Diffix is distributed as $\mathcal{N}(C + I\{r^u_{a_n} = 0\}, 4)$, assuming the query is thus not bucket suppressed (which will occur only if $r^u_{a_1}$ is very rare). Repeating this query for different attributes $a_2, \ldots, a_{n-1}$ thus gives n samples to distinguish between (in effect) $\mathcal{N}(1, 4)$ and $\mathcal{N}(0, 4)$. For $n = 5$, a likelihood ratio test distinguishing between these distributions can already achieve an accuracy of $\approx 73\%$.

4.6.3 Non-deterministic systems: SimpleQBS

We show how QuerySnout is also able to find efficient attacks against non-deterministic systems. While we here show and discuss results against the Adult dataset, our conclusions hold for Census and Insurance (cf. Fig. 4.5). We also compare our results to the baseline inspired by Denning et al. [97] and using a simple heuristic to choose difference queries from the auxiliary information available to the adversary (see Sec. 4.5.4 for details).

Fig. 4.4 shows how QuerySnout manages to find effective attacks against SimpleQBS across a range of σ (noise) and τ (threshold) outperforming, often strongly, manual attacks. Specifically, QuerySnout outperforms–on average–manual attacks in the AUXILIARY scenario by 3.8%, particularly when the noise added is small (σ from 0 to 3). The gap in accuracy is even larger (8.4% on average) in the EXACT-BUT-ONE scenario, finding attacks with $> 90\%$ accuracy

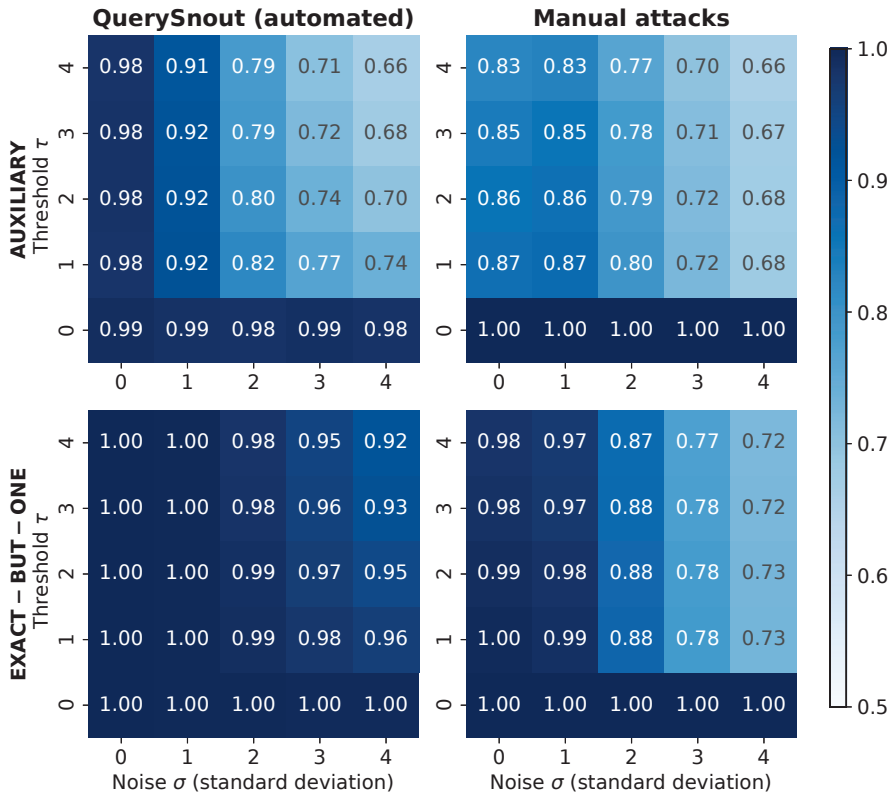
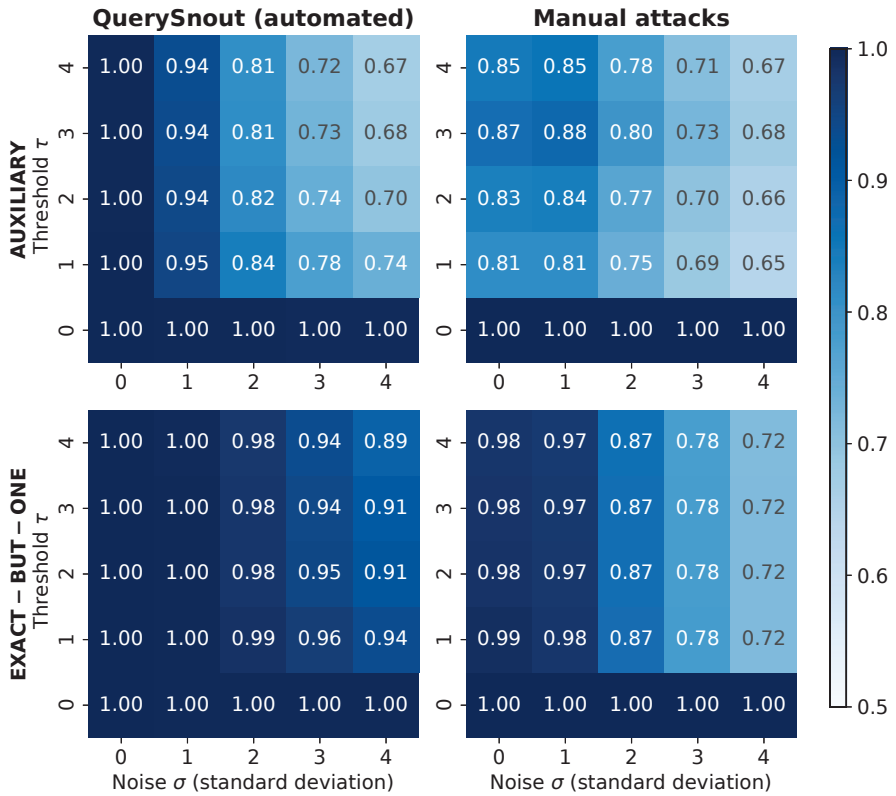
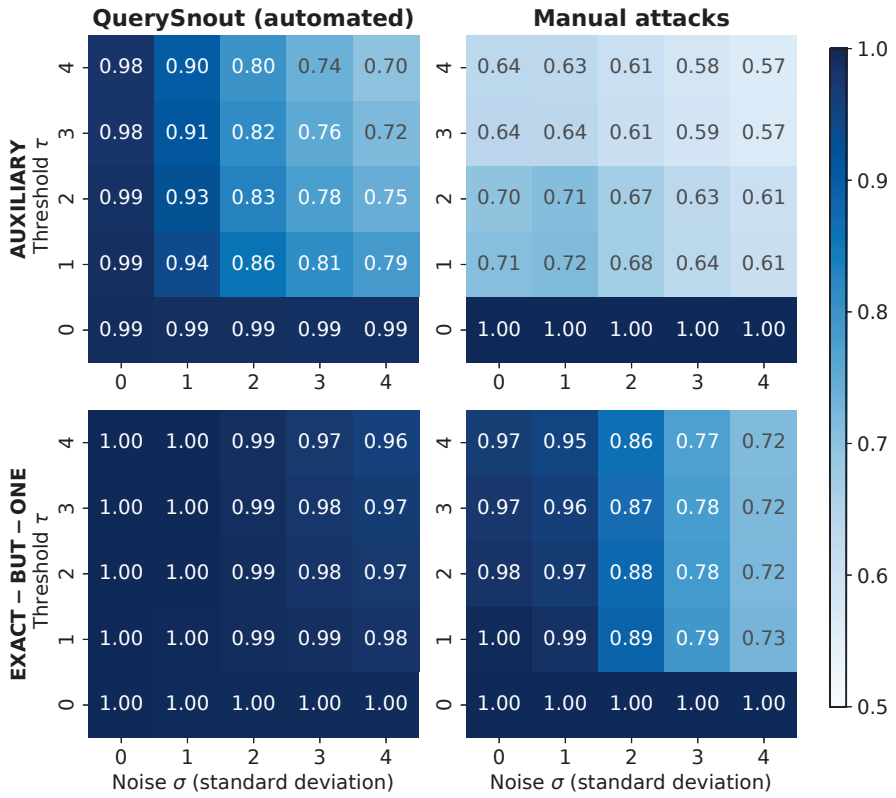


Figure 4.4: **Comparison between automated and manual attacks against SimpleQBS(τ, σ) on the Adult dataset.** We report the accuracy of attribute inference attacks discovered by QuerySnout (left) and of manual attacks tailored to each system (right) in the AUXILIARY (top) and EXACT-BUT-ONE (bottom) scenarios. We report the attack accuracy over 100 target records, averaged over 5 repetitions.

across all values of (τ, σ) considered. Noticeably, QuerySnout finds much better attacks, by 16.7% on average, than the manual ones for high values of σ (from 2 to 4) and $\tau \geq 1$.



(a) Census



(b) Insurance

Figure 4.5: **Comparison between automated and manual attacks against SimpleQBS(τ, σ) on other datasets.** We report the attack accuracy over 100 target records, averaged over 5 repetitions.

4.6.4 Comparison with random search

To evaluate the impact of our evolutionary search procedure (i.e., mutations), we compare our results with the accuracy obtained by a random search. The random search uses the same fitness evaluation but samples a new population of random solutions at every generation rather than mutating and keeping previous solutions. It returns the best solution found over all generations. Algorithm 6 details the random search procedure. We here compare our approach to random search for Diffix, Table Builder, and two selected version of SimpleQBS: ($\tau = 4, \sigma = 3$) and ($\tau = 3, \sigma = 4$). We find that our approach strongly improves upon a random search procedure using the same parameters, across all datasets and QBSs (see Table 4.3). For instance, in the AUXILIARY scenario, our approach outperforms the random search by 6.3% for Diffix, 5.3% for TableBuilder, 6.7% for SimpleQBS($\tau = 4, \sigma = 3$), and 5.4% for SimpleQBS($\tau = 3, \sigma = 4$), each time averaged over the datasets.

Algorithm 6 RandomSearch

Inputs:
P: Population size (number of solutions).
m: Number of queries in a solution.
N: Number of generations.
D_{aux}: Auxiliary datasets used to evaluate the fitness.
Output:
$best \in \mathcal{S}^m$: the best solution found in the search.
Initialise:
$best \leftarrow None$
$best\_fitness \leftarrow -\infty$
// Iterate over N generations.
for $g = 1$ to N **do**
  // Generate a random population.
  for $i = 1$ to P **do**
    $solution \leftarrow random\_solution(m)$
    $fit \leftarrow$ EvaluateFitness($solution, D_{\text{aux}}, U_{\text{aux}}$)
    if $fit > best\_fitness$ **then**
      $best\_fitness \leftarrow fit$
      $best \leftarrow solution$
    end if
  end for
end for

Table 4.3: **Comparison between QuerySnout and the random search.** We report the absolute difference (in percentage points) between the accuracy of QuerySnout and the accuracy of the random search (with mean and standard deviation over 5 repetitions).

	Diffix			**TableBuilder**		
	Adult	**Census**	**Insurance**	**Adult**	**Census**	**Insurance**
AUXILIARY	6.8 (0.4)	5.6 (0.8)	6.5 (0.7)	6.0 (0.3)	4.1 (1.1)	5.9 (1.5)
EXACT-BUT-ONE	4.8 (0.3)	5.0 (0.3)	4.7 (0.3)	2.9 (1.0)	3.3 (0.8)	1.3 (0.5)

	SimpleQBS($\tau = 4, \sigma = 3$)			**SimpleQBS($\tau = 3, \sigma = 4$)**		
	Adult	**Census**	**Insurance**	**Adult**	**Census**	**Insurance**
AUXILIARY	6.4 (0.4)	7.0 (0.4)	6.7 (0.3)	5.1 (0.4)	4.9 (0.9)	6.2 (0.4)
EXACT-BUT-ONE	7.5 (2.5)	10.3 (2.2)	4.1 (1.7)	6.7 (1.7)	8.4 (2.0)	3.4 (1.8)

4.6.5 Impact of the number of queries

Limiting the number of queries is, along with authentication, a popular "non-QBS" defence used in practice. This can be done directly, often per user and per period of time, or through budgeting [213]. We here evaluate the impact of limiting or increasing the number of queries made by our attack to the target QBS. For simplicity, we report results on a deterministic QBS (Diffix) and a non-deterministic QBS (SimpleQBS($\tau = 4, \sigma = 3$)) on the Adult dataset in the AUXILIARY scenario. The τ and σ of SimpleQBS were chosen to be comparable to those of Diffix, same threshold mean and similar average noise per query (for queries with 5 conditions).

Fig. 4.6 shows that, as expected, the performance of QuerySnout increases with the number of queries m, albeit slowly. Indeed, even with as little as 10 queries, QuerySnout still reaches an accuracy of 69.3% against Diffix and 64.4% against SimpleQBS(4, 3). Increasing the number of queries would further increase the accuracy of the attack, e.g., against systems that do no implement some kind of query limiting mechanisms.

We also evaluate the impact of m on both random search and a naïve baseline consisting of one solution sampled uniformly at random from the search space of solutions $\mathcal{S}^m$. Our results show that QuerySnout's mutation operators enable it to consistently strongly outperform the random search, which itself –as expected– outperforms a random solution. It is however interesting to note that, when the number of queries is large, even a single random solution achieves better

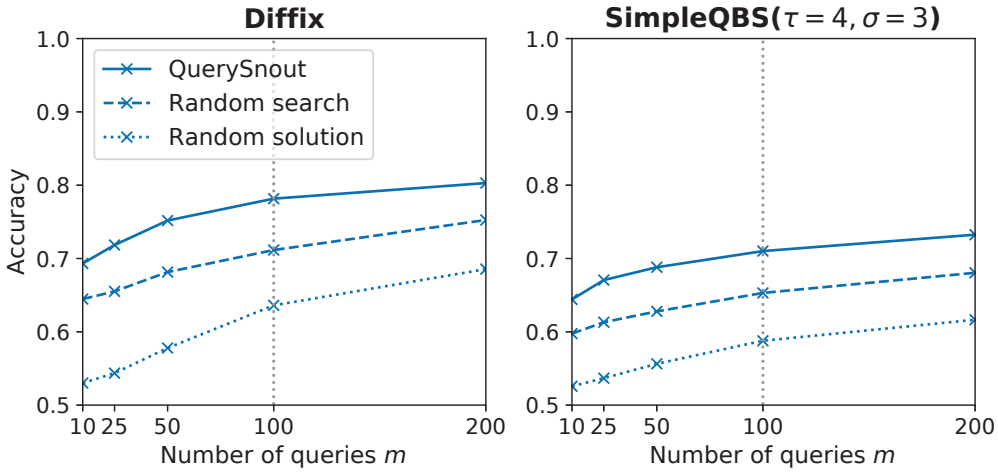


Figure 4.6: **Impact of the number of queries on the accuracy of QuerySnout.** We report the accuracy of QuerySnout (which uses an evolutionary search), of a random search, and of a solution chosen uniformly at random as we vary the number of queries m. The results are computed on the Adult dataset in the AUXILIARY scenario.

than random accuracy.

4.6.6 Impact of search parameters

We study the impact of different components on the performance of QuerySnout. Fig. 4.7 shows that increasing the size of the population P, the number of generations N, or the number of datasets (auxiliary QBSs) results in better performance, with decreasing returns. However, the marginal increase in performance comes at a computational cost, as doubling any of the parameters also doubles the running time. In some cases, the memory footprint also increases significantly, e.g., increasing linearly with the number of auxiliary datasets.

For the mutation operators, Table 4.4 shows "copy" to be the most impactful one. For non-deterministic systems, "copy" duplicates a query and – with equal probability – either keeps the duplicate unchanged or modifies it with p_{change} and p_{swap}. For deterministic systems, "copy" duplicates a query and always modifies it, since repeating the same query would yield the same answer. In both cases, removing this component negatively impacts the evolutionary search by a significant margin. Even though queries are still being modified (according to p_{modify}), it becomes harder for QuerySnout to find difference queries or to repeat good queries many

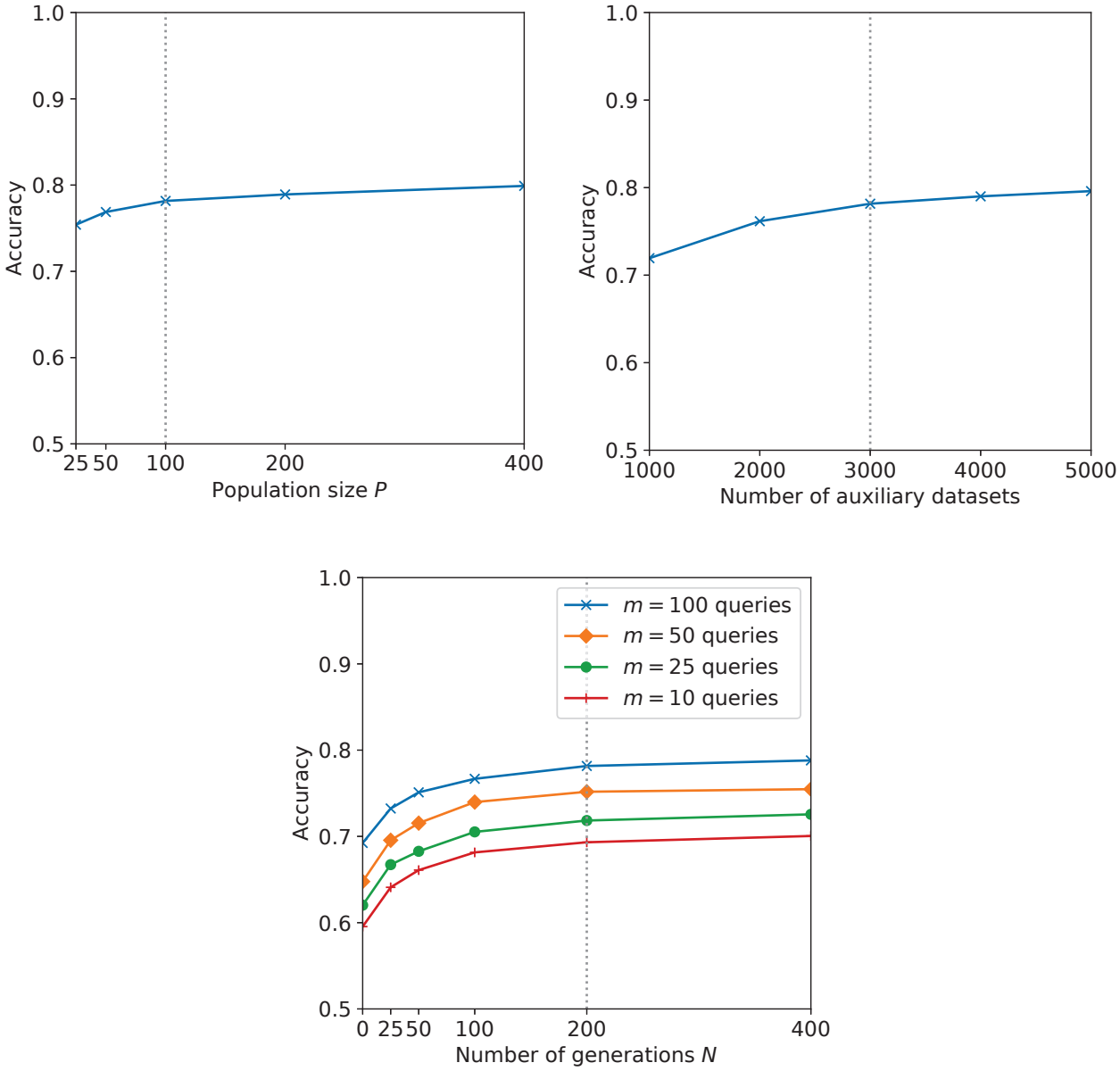


Figure 4.7: **Impact of various search parameters on the accuracy of QuerySnout.** The results are computed on the Adult dataset in the AUXILIARY scenario. The vertical gray line indicates the default value used in the main experiment.

times. As for the mutations of query operators, the search is robust to removing either "swap" or "change", as there is significant overlap between the two mutations. We further hypothesise that "swap" is likely to be more useful when increasing the number of condition operators $|\mathcal{C}_s|$ (e.g., allowing for $<, \leq, \geq, >$ operators).

Table 4.4: **Ablation experiment in the AUXILIARY scenario.** We report the accuracy of the evolutionary search (with mean and standard deviation over 5 repetitions) where we set one of the mutation probabilities to zero, keeping all the others equal to the default value.

	Deterministic systems		**Non-deterministic systems**		
	Diffix	**TableBuilder**	**SimpleQBS** ($\tau = 4, \sigma = 3$)	**DPLaplace** ($\varepsilon = 1$)	**DPLaplace** ($\varepsilon = 10$)
Default values	77.8 (0.5)	84.5 (0.6)	70.7 (1.1)	62.9 (0.8)	98.6 (0.4)
$p_{\text{copy}} = 0$	76.1 (0.5)	82.9 (0.3)	67.2 (1.3)	52.1 (0.3)	93.0 (0.3)
$p_{\text{change}} = 0$	76.1 (0.7)	83.5 (0.5)	70.5 (1.2)	65.5 (1.0)	98.5 (0.4)
$p_{\text{swap}} = 0$	77.7 (0.6)	84.5 (0.6)	71.0 (1.1)	63.4 (0.8)	98.5 (0.4)

4.6.7 Impact of dataset size

Fig. 4.8 shows that the accuracy of attacks discovered by QuerySnout slowly increases as the dataset $|D|$ decreases (in the AUXILIARY scenario). This can partly explain why our attack performs better on Insurance ($|D| = 1000$) than on Adult ($|D| = 8000$). We believe that this might be due to partitions of the original dataset being of limited size (e.g., slightly larger than 16K for Adult), so that large datasets sampled from, e.g., π_{train} tend to overlap more and hence generalise less well. In our experiments, we found the validation accuracy to remain the same or slightly decrease with $|D|$, while the gap between the validation and test accuracy slightly increases. Note that these results were conducted using only one seed, for computational reasons. We leave the thorough investigation of reasons why this occurs for future work.

4.6.8 Analysis of solutions

We analyse the solutions found by QuerySnout against Diffix and TableBuilder in the AUXILIARY scenario. First, we automatically identify and extract the difference queries from each solution (corresponding to a target record) and repetition. Fig. 4.9a shows that out of $m = 100$ queries, QuerySnout uses an average of 29.0 difference queries on Adult, 26.9 on Census and 29.0 on Insurance. We then fit a logistic regression on answers to these queries using the same seed and hyperparameters as in the default experiment. Fig. 4.10a shows that these queries account for 98.0% of the accuracy on Adult and for 97.0% of the accuracy on Census and

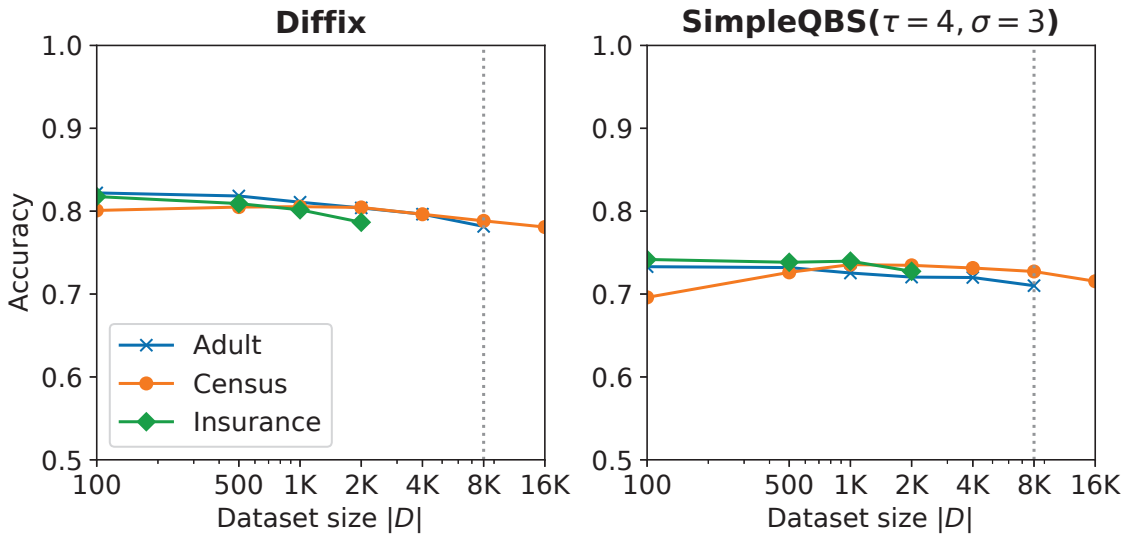
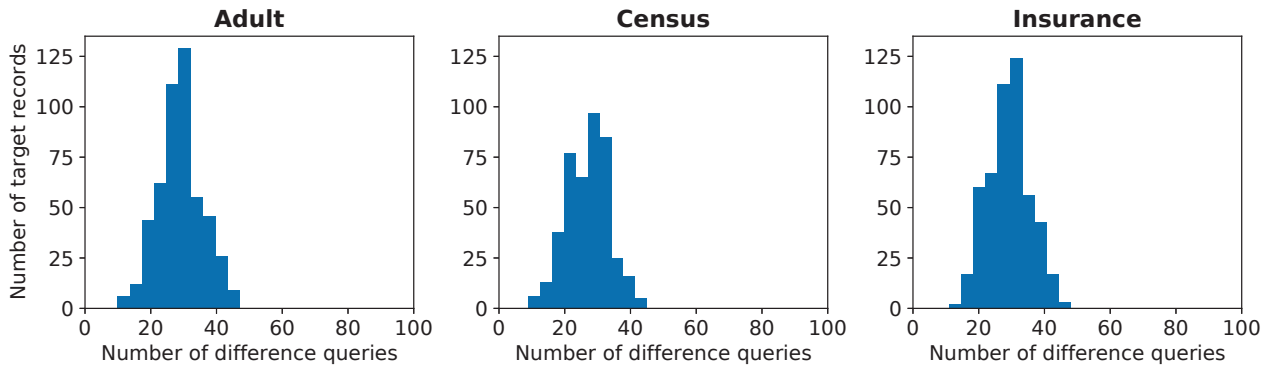


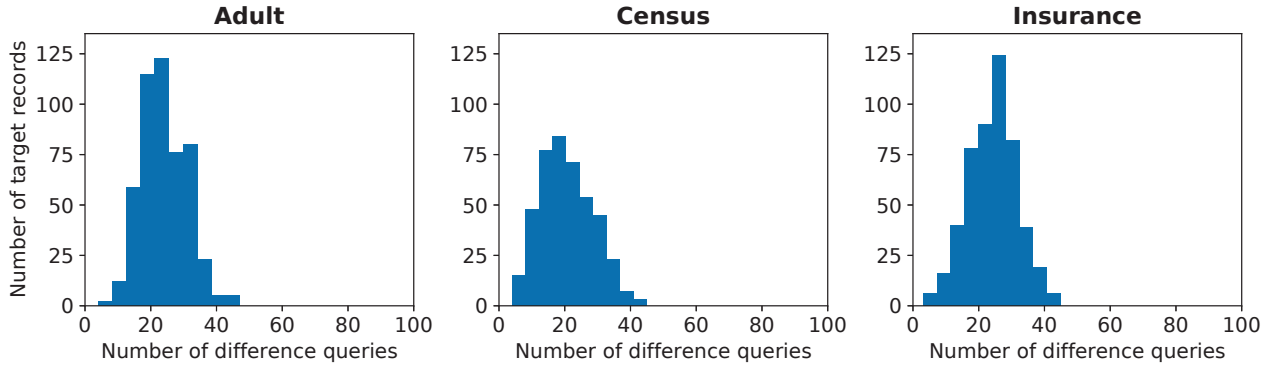
Figure 4.8: **Impact of the dataset size on the performance of QuerySnout.** We show the accuracy of an attribute inference attack discovered by QuerySnout in the AUXILIARY scenario, for varying dataset sizes. The dotted vertical line represents the value 8000 which we use for experiments on Adult and Census.

Insurance. This is much less compared to using a random subset of each solution of the same size, which only preserves, on average, 80.0% of the performance on Adult, 79.8% on Census, and 80.5% on Insurance.

We obtain similar results on TableBuilder, shown on Fig. 4.9b and Fig. 4.10b. Specifically, QuerySnout finds fewer difference queries (an average of 23.9 on Adult, 20.7 on Census, and 24.2 on Insurance), but those found account for most of the performance: 99.8% on Adult, 97.5% on Census, and 98.2% on Insurance. The randomly selected queries only account for 73.1% of the performance on Adult, 71.7% on Census, and 74.8% on Insurance.

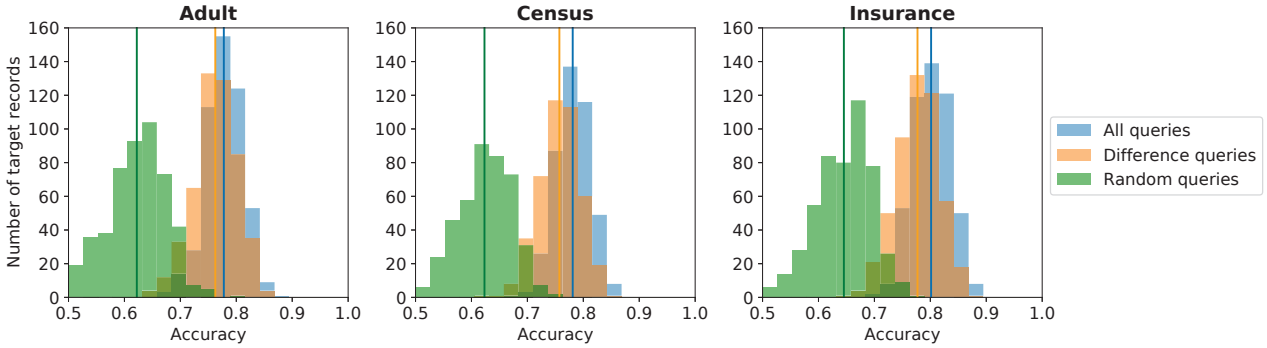


(a) Diffix

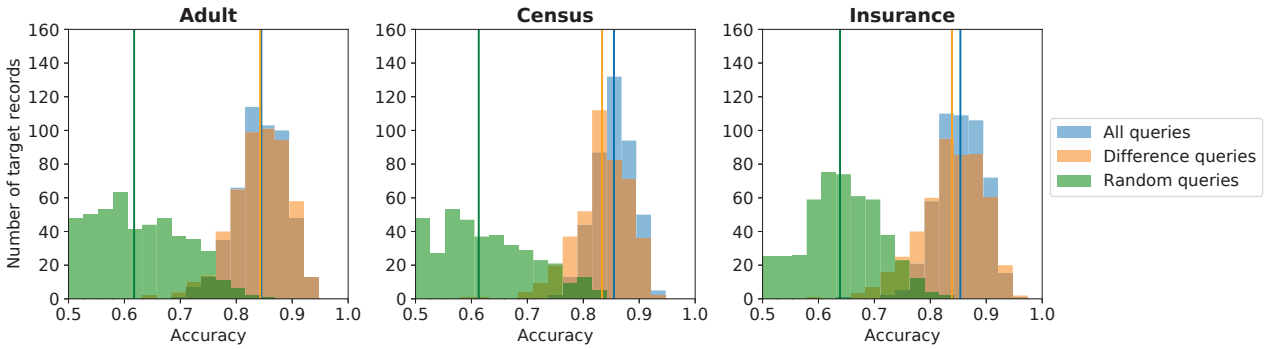


(b) TableBuilder

Figure 4.9: **Histogram of the number of difference queries in each solution found by QuerySnout in the AUXILIARY scenario.** Histograms are computed over all solutions from the 5 repetitions.



(a) Diffix



(b) TableBuilder

Figure 4.10: **Performance of attacks that use all the queries in the solution found by QuerySnout (blue), only the difference queries (orange), and the same number of random queries in the solution (green).** Histograms are computed over all solutions from the 5 repetitions. The vertical bars show the mean accuracy.

4.7 Budget-based systems

Some QBSs, mostly based on $\varepsilon-$differential privacy [107] (described in Sec. 2.2.2), require the user to divide a given privacy budget ε between the queries they ask. We here present a simple heuristic to apply QuerySnout to QBSs that require a budget, which we call Budget-Based Systems (BBS).

4.7.1 Extending QuerySnout to BBSs

Formally, we model budget-based systems similarly to other QBSs, with the addition of a parameter called the *partial budget* of the query, specifying the fraction of the total budget (ε) to use on this query: $R : \mathcal{D} \times \mathcal{Q} \times (0, 1]$. A new query is only answered if the sum of partial budgets used and of the new query's partial budget is less than or equal to 1.

We here propose a simple heuristic to transform the multiset of queries manipulated by QuerySnout to a list of pairs of query and partial budget. Given a solution of m queries $s = (c^1, \ldots, c^m)$ we group identical queries to obtain k unique queries with multiplicities $(c^{i_1}, m_1), \ldots, (c^{i_k}, m_k)$ such that $\sum_{i=1}^{k} m_i = m$. We allocate to each unique query a partial budget proportional to its multiplicity. This way, we perform k queries to the QBS, where the partial budget allocated to the i-th query is equal to $\frac{m_i}{m}$. This heuristic assumes *monotonicity of accuracy*: performing any query q with full budget yields more accurate results than repeating a query k times with a partial budget of $1/k$ and averaging out the results.

Note that while running the evolutionary search, we are performing the queries on QBSs instantiated on the auxiliary datasets (resetting their budget at each iteration), and only the final $k \leq m$ queries will be performed on the target QBS. Hence the budget of the target QBS is only used once.

4.7.2 DPLaplace

We consider a simple budget-based mechanism using addition of Laplace noise. The mechanism assumes that a total budget ε is allocated by the data curator, and that each query is answered by adding independent Laplace noise scaled with $(p\varepsilon)^{-1}$, where p is the fractional budget allocated for this query. This ensures by theorem 2.2.2 that each answer satisfies $(p\varepsilon)-$Differential Privacy (DP) and, by composition [213], performing k queries with fractional budgets $(p_1, \ldots, p_k)$ satisfies $(\sum_{i=1}^{k} p_i\varepsilon)-$DP. Formally, having answered $k-1$ queries with total fractional budget $p_{1:k-1}$, the QBS answers the kth query with fractional budget p_k iff $p_{1:k-1} + p_k \leq 1$, and answers as:

$$R_{\text{DPLaplace}(\varepsilon)}(D, Q, p) = T(D, Q) + L \quad \text{with } L \sim Lap\left((p\,\varepsilon)^{-1}\right) \tag{4.11}$$

Similarly to the other systems, we round answers to the nearest integer and threshold the answers at 0, which does not affect the privacy guarantees. Although the DPLaplace mechanism is particularly simple, it serves as the core component of many more complex systems implementing Differential Privacy, such as PINQ [213], Chorus [232] and PriPearl [177]. We instantiate this QBS for $\varepsilon \in \{1, 5, 10\}$. As we explain below, finding solutions for our search procedure is challenging against DPLaplace.

We now show that the DPLaplace mechanism satisfies the monotonicity of accuracy property. Formally, a mechanism R satisfies monotonicity of accuracy if and only if for all $q \in \mathcal{Q}$, $d \in \mathcal{D}$, and $k \in \mathbb{N}, k > 1$ (with error $e : r \mapsto (r - T(d, q))^2$):

$$\mathbb{E}\left[e\left(R(d, q, 1)\right)\right] \leq \mathbb{E}\left[e\left(\sum_{i=1}^{k} \frac{R(d, q, 1/k)}{k}\right)\right] \tag{4.12}$$

Note that this is a desirable property of QBS based on budget: if a QBS is non-monotonous, analysts are incentivised to repeat the same query as many times with as small a budget as possible, rather than perform queries directly, which is impractical.

Theorem 4.1. *The $R_{DPLaplace}$ be the DPLaplace mechanism satisfies monotonicity of accuracy*

for all budgets $\varepsilon > 0$, i.e., for all $q \in \mathbb{Q}$, $d \in \mathcal{D}$, and $k \in \mathbb{N}, k > 2$:

$$\mathbb{E}\left[e\left(R_{DPLaplace(\varepsilon)}(d,q,1)\right)\right] \leq \mathbb{E}\left[e\left(\sum_{i=1}^{k} \frac{R_{DPLaplace(\varepsilon)}(d,q,1/k)}{k}\right)\right] \tag{4.13}$$

where $e : r \mapsto (r - T(d,q))^2$ is the $\mathbb{L}_2$ error.

Proof. Define $L = R_{\text{DPLaplace}(\varepsilon)}(d,q,1) - T(d,q)$. By definition, observe that $L \sim Lap(\frac{1}{\varepsilon})$. By definition, we have that:

$$\mathbb{E}\left[e\left(R_{\text{DPLaplace}(\varepsilon)}(d,q,1)\right)\right] = \mathbb{E}\left[L^2\right] = \frac{2}{\varepsilon^2} \tag{4.14}$$

Similarly, define $L_i = R_{\text{DPLaplace}(\varepsilon)}(d,q,1/k) - T(d,q)$ for $i = 1,\ldots,k$, and observe that $L_i \sim Lap(\frac{k}{\varepsilon})$. We have that:

$$\mathbb{E}\left[e\left(\sum_{i=1}^{k} \frac{R_{\text{DPLaplace}(\varepsilon)}(d,q,1/k)}{k}\right)\right] = \mathbb{E}\left[\left(\frac{\sum_{i=1}^{k} L_i}{k}\right)^2\right] \tag{4.15}$$

By linearity of the expectation:

$$\mathbb{E}\left[\left(\frac{\sum_{i=1}^{k} L_i}{k}\right)^2\right] = \frac{1}{k^2} \cdot \sum_{i=1}^{k} \mathbb{E}[L_i^2] = \frac{1}{k}\frac{2k^2}{\varepsilon^2} = \frac{2k}{\varepsilon^2} > \frac{2}{\varepsilon^2}, \tag{4.16}$$

which concludes the proof. □

Known optimal attack. A known *optimal* attack exists against DPLaplace: a uniqueness attack that performs the query:

$$q = \text{COUNT WHERE} \bigwedge_{i=0}^{n-1} (a_i = r_{a_i}^u) \wedge a_n = 0 \tag{4.17}$$

with partial budget 1 and returns $s = 0$ iff the count is larger than 0.5. We prove below that this attack achieves maximal accuracy. Given a number of queries m, this attack can be mapped to the solution in $\mathcal{S}^m$ consisting of the query q repeated m times. While the attack is simple for a knowledgeable adversary, it present challenges for our search procedure. Showing that

QuerySnout can be extended to budget-based mechanisms is therefore important.

The uniqueness query of Eq. 4.17 reveals the target user's secret attribute with some added noise:

$$R := R(d, q^*, p) = I\{r_n^u = 1\} + L, \quad L \sim Lap(\frac{1}{p\varepsilon}). \tag{4.18}$$

Because the QBS has monotonicity of accuracy, the best choice of p possible is $p = 1$, i.e. using all the budget for this query. Hence, $R \sim Lap(0, \varepsilon^{-1})$ if the user's target record is 0, and $R \sim Lap(1, \varepsilon^{-1})$ otherwise.We can thus use a likelihood ratio test to determine the target record's value. In the continuous case, this gives the rule:

$$\widehat{r_n^u = 1} \iff \frac{pdf_{Lap(1,\varepsilon^{-1})}(R)}{pdf_{Lap(0,\varepsilon^{-1})}(R)} \geq 1 \iff R \geq \frac{1}{2}, \tag{4.19}$$

For the rounded and thresholded at zero QBS that we report results on, the same rule applies. Indeed, let $P_{\mathcal{L}}(n)$ the probability of sampling $n \in \mathbb{N}$ from rounding and thresholding of a sample of $L \sim \mathcal{L}$. By developing the pdf of the Laplace distribution and integrating over bins, we get:

- $P_{Lap(0,\varepsilon^{-1})}(0) = \frac{1}{2} + \int_0^{\frac{1}{2}} \frac{\varepsilon}{2} e^{-x\cdot\varepsilon} dx > \frac{1}{2}$,
- $P_{Lap(1,\varepsilon^{-1})}(0) = \int_{-\infty}^{\frac{1}{2}} \frac{\varepsilon}{2} e^{(x-1)\cdot\varepsilon} dx < \frac{1}{2}$,
- $\forall r \in \mathbb{N}_0 : \; P_{Lap(0,\varepsilon^{-1})}(r) = \int_{r-\frac{1}{2}}^{r+\frac{1}{2}} \frac{\varepsilon}{2} e^{-x\cdot\varepsilon} dx$,
- $\begin{aligned} P_{Lap(1,\varepsilon^{-1})}(1) &= \int_{\frac{1}{2}}^{1} \frac{\varepsilon}{2} e^{(x-1)\cdot\varepsilon} dx + \int_{1}^{\frac{3}{2}} \frac{\varepsilon}{2} e^{(1-x)\cdot\varepsilon} dx \\ &= P_{Lap(0,\varepsilon^{-1})}(1) + f(\varepsilon) \end{aligned}$, where $f(\varepsilon) = 1 - \frac{3}{2}e^{-\frac{\varepsilon}{2}} + \frac{1}{2}e^{-\frac{3\varepsilon}{2}} > 0, \quad \forall \varepsilon > 0$ (since $f(0) = 0$ and $f'(\varepsilon) > 0$ for $\varepsilon > 0$),
- $\forall r \in \mathbb{N} \setminus \{0, 1\} : \; P_{Lap(1,\varepsilon^{-1})}(r) = \int_{r-\frac{1}{2}}^{r+\frac{1}{2}} \frac{\varepsilon}{2} e^{(1-x)\cdot\varepsilon} dx = e^{\varepsilon} P_{Lap(0,\varepsilon^{-1})}(r) > P_{Lap(0,\varepsilon^{-1})}(r)$.

We thus observe that for all $n \in \mathbb{N}$, $P_{Lap(0,\varepsilon^{-1})}(n) \geq P_{Lap(1,\varepsilon^{-1})}(n)$ iff $n = 0$. These tests have maximum statistical power, thanks to the Neyman-Pearson lemma [235].

Search challenges. Finding good solutions with our current black-box search is challenging here, particularly for small values of ε. Indeed, a query that is not repeated (equivalently, that has a small partial budget) will be answered with a noise of large standard deviation ≈ 14 for

$\varepsilon = 1$ and $m = 10$. This has two consequences: first, estimating the fitness of solutions is difficult (due to the increased randomness), which can lead to overfitting, both of the rule G and the fitness; second, it complicates the discovery of good queries, since the signal of optimal queries is significantly smaller than the noise added. For instance, even the optimal query $\bigwedge_{i=1}^{n-1}(a_i = r^u_{a_i}) \wedge s = 0$ has little signal if not repeated, since performing it once (with budget $\frac{1}{10}$) leads to an attack with accuracy of at most $\approx 52.4\%$.

Attack parameters. Unlike the previous mechanisms, increasing the number of queries in a solution m amounts to adding more noise to the answers. The standard deviation of the noise is indeed proportional to m, due to the budget being split among the queries (and is equal to $\sqrt{2}m/\varepsilon$ for queries with multiplicity 1). We thus use smaller solutions of $m = 10$ queries. All other parameters are identical as those in Sec. 4.6.1, except we update the mutation operators accordingly to $p_{\text{copy}} = p_{\text{modify}} = \frac{1}{m} = 0.1$.

4.7.3 Empirical results

Table 4.5 shows that the attacks discovered by QuerySnout match the performance of the optimal attack for large budget values $\varepsilon \in \{5, 10\}$ but fall slightly short in the $\varepsilon = 1$ case. Indeed, in the latter case QuerySnout performs on average 6.4% worse than the optimal attack for AUXILIARY, and 4.1% for EXACT-BUT-ONE.

These results show that QuerySnout can already find the optimal attacks against budget-based mechanisms for medium to small amounts of noise, and very good ones when a large amount of noise is added. We believe the search can be further improved in future work.

Table 4.6 shows that on DPLaplace, QuerySnout vastly outperforms the random search for all datasets, scenarios, and values of ε. Specifically, in the AUXILIARY scenario, the gap is of 11.5% on average across the datasets for $\varepsilon = 1$, with the random search (at 52.1%) barely improving on the random guess baseline. Similar or higher gaps (up to 29.3%) are obtained for other datasets, scenarios, and values of ε. This result both confirms that finding good solutions for DPLaplace is challenging, and shows that QuerySnout is able to efficiently explore

Table 4.5: **Accuracy against DPLaplace for different values of ε.** We report the mean and standard deviation over 5 repetitions.

	DPLaplace($\varepsilon = 1$)			**DPLaplace($\varepsilon = 5$)**		
AUXILIARY	**Adult**	**Census**	**Insurance**	**Adult**	**Census**	**Insurance**
QuerySnout (automated)	62.9 (0.8)	64.2 (0.8)	63.6 (0.8)	94.2 (1.0)	94.6 (1.0)	94.9 (0.7)
Uniqueness attack (manual)	69.2 (0.8)	70.2 (0.9)	70.4 (1.2)	95.6 (0.8)	95.5 (0.7)	95.9 (0.6)
EXACT-BUT-ONE	**Adult**	**Census**	**Insurance**	**Adult**	**Census**	**Insurance**
QuerySnout (automated)	64.2 (0.7)	65.0 (0.6)	65.6 (0.8)	95.4 (0.8)	95.3 (0.7)	95.7 (0.7)
Uniqueness attack (manual)	68.6 (1.1)	68.9 (2.6)	69.5 (2.6)	95.7 (1.1)	95.6 (1.1)	96.0 (0.6)

	DPLaplace($\varepsilon = 10$)		
AUXILIARY	**Adult**	**Census**	**Insurance**
QuerySnout (automated)	98.6 (0.4)	99.4 (0.2)	99.0 (0.3)
Uniqueness attack (manual)	99.6 (0.2)	99.6 (0.2)	99.7 (0.1)
EXACT-BUT-ONE	**Adult**	**Census**	**Insurance**
QuerySnout (automated)	99.5 (0.3)	99.6 (0.1)	99.7 (0.1)
Uniqueness attack (manual)	99.5 (0.3)	99.5 (0.4)	99.8 (0.2)

Table 4.6: **Comparison between QuerySnout and the random search on DPLaplace for different values of ε.** We report the absolute difference (in percentage points) between the accuracy of QuerySnout and the accuracy of the random search (with mean and standard deviation over 5 repetitions).

	DPLaplace($\varepsilon = 1$)			**DPLaplace($\varepsilon = 5$)**		
	Adult	**Census**	**Insurance**	**Adult**	**Census**	**Insurance**
AUXILIARY	10.8 (0.6)	11.9 (0.7)	11.4 (0.9)	26.0 (0.7)	25.8 (0.8)	26.5 (0.4)
EXACT-BUT-ONE	11.5 (0.8)	12.2 (0.6)	12.9 (1.0)	29.1 (1.0)	28.6 (1.1)	29.3 (1.2)

	DPLaplace($\varepsilon = 10$)		
	Adult	**Census**	**Insurance**
AUXILIARY	17.7 (0.5)	17.6 (0.8)	17.9 (0.6)
EXACT-BUT-ONE	16.1 (0.5)	15.6 (0.3)	15.8 (0.5)

the search space even in complex cases.

Overall, the gap between QuerySnout and the random search is much larger for DPLaplace compared to the other QBSs considered in this work. This is due to (1) random solutions yielding low accuracy against DPLaplace (as explained above) and (2) the noise added by the other QBSs to answers on one query being overall smaller ($\leq \sqrt{12}$ for Diffix and $\sigma \leq 5$ for SimpleQBS, compared to $10\sqrt{2}$ for DPLaplace($\varepsilon = 1$)) and independent of the number of

queries.

4.8 Discussion

QuerySnout *automatically* finds attacks that equate and often outperform previous manual attacks across a range of systems and datasets. This demonstrates the potential of automating attacks against QBSs and opens the door for new, much more complex attacks to be discovered. Our work is the first to enable systems to be automatically tested before deployment to identify and patch vulnerabilities. We now discuss in more detail the extensive potential avenues for future work.

Extensions. QuerySnout currently operates in the same constrained space explored by manual attacks. More complex and accurate attacks exploiting e.g. different SQL operators ($\leq, \geq$) or logical expressions (such as OR) are, in our opinion, likely to exist. Finding such attacks however presents some challenges as the size of the search space increases fast with new operators or conditions per attributes, and with it the computational cost of the search. Specialised mutation operators can be used to overcome this challenge, for instance, using empirical discrete gradient approximation [103]. Furthermore, generative encodings [314] can be used to improve the expressiveness of the genotype and enhance the capability of QuerySnout to find regularities in query sets.

QuerySnout currently relies on a set of assumptions, primarily that the target record is unique in the private dataset and that the attacker has access to auxiliary knowledge on the target dataset. We believe both can be relaxed. In particular, different auxiliary information can likely be used, for instance a few statistics on the data from which synthetic data is generated [302], or syntactically similar datasets from another distribution. These would not require changing the method as QuerySnout only requires two samplers π_{train} and π_{val}. The extent to which using different auxiliary information (e.g., computing the query answers used to train the classifier G on synthetic datasets) affects the attack performance is an interesting open question.

The attacks that we consider here are *non-interactive*: the attacker chooses which queries to

perform *before* observing answers from the target QBS. It is likely that attacks could be made more accurate or efficient if future automated attacks were to use the answer to previous queries to inform the choice of the next queries [129].

Finally, QuerySnout could be extended to use additional QBS outputs such as the time it took for the answer to a query to be computed (such as exploited by Boenisch et al. [41]), information about the query provided by the QBS, or warnings and error messages produced by the QBS (e.g., allowing our system to treat bucket suppression as different from a 0). This would enable the system to discover so-called *side-channel attacks* in addition to the privacy attacks that this work focuses on.

Membership inference attacks. QuerySnout currently focuses on attribute inference attacks. Another class of attacks, membership inference attacks (MIA), has been considered in the privacy literature. MIAs are at the core of formal privacy guarantees such as differential privacy [107] and can lead to privacy concerns when being part of a dataset is sensitive. More importantly here, membership inference can also be a first step towards attribute inference attacks by allowing the attacker to verify the assumption that the target record is in the dataset. QuerySnout can be extended to membership inference attacks with minimal changes. Indeed, the main change to our method is in the sampling of auxiliary datasets used to train the attack: after sampling a dataset (as a subset of D' in the AUXILIARY scenario, or as D in the EXACT-BUT-ONE), with equal probability either a random record in the dataset is replaced with the target record, or the auxiliary dataset left unchanged. The rule G can then be trained to predict membership (whether the target record is in the data) rather than the value of the sensitive attribute. The rest of the method can be applied unchanged, although it is likely that designing specific mutations for membership inference might result in better accuracy.

Automating the analysis of solutions. In this work, we manually inspected the queries found by QuerySnout to understand what vulnerabilities of the QBS they exploited. This analysis could likely be automated in the future, e.g. using techniques from interpretability in machine learning [275] to understand the relative importance of queries in the attack. This

would greatly help better understand what the attack exploits and how the attack could be mitigated and patched to make the system stronger.

Automating the defence. QuerySnout discovers highly accurate attacks against real-world systems in the AUXILIARY scenario, and new, more powerful, attacks in the EXACT-BUT-ONE scenario. We believe that defending against these attacks will likely require both improving these systems but also ensuring that appropriate risk mitigation strategies are in place, including authentication, query limits, and logging. We believe however that the existence of an automated attack system such as QuerySnout might help develop stronger mitigation strategies such as anomaly detection and attack detection mechanisms.

Order of queries. We here assume that the order of queries does not matter to the QBS (Sec 4.4.2), a property satisified by all the query-based systems we consider, as well as, in our opinion, the majority of real-world systems. In some cases however, this might not be the case. For instance, one could imagine a system auditing queries in order to avoid answering the second half of difference query pairs [225]. Such a QBS could be represented by a function $R_k : \mathcal{D} \times \mathcal{Q} \times \mathcal{H} \to \mathbb{R}$, where $\mathcal{H}$ is the set of *histories*, i.e., lists of query-answer pairs. Here, solutions would be represented as *ordered* lists of queries rather than multisets, which increases the size of the solution search space by a factor of approximately $m!/3^n$. Our method can be adapted by adding a mutation operator which changes the query order.

Applicability to other systems. QuerySnout can be applied, in all generality, to any QBS providing answers to a minimal syntax consisting of the counting queries described in Eq. 4.2.1. This is because the machine learning classifier only requires answers to m queries computed on different datasets in order to be trained, and the evolutionary search is designed to explore the query search space described in Sec. 4.4.2, which is a subset of the minimal syntax. We evaluated QuerySnout and showed it to be effective against a broad range of query-based systems, including real-world mechanisms such as Diffix [123] and TableBuilder [246] and budget-based mechanisms implementing DP guarantees [107]. Furthermore, we considered both deterministic and non-deterministic systems, with noise distributions including the Gaussian (SimpleQBS), Uniform (TableBuilder), and Laplace (DPLaplace) distributions.

However, our method may not be effective against every system. For instance, while we showed it to be effective against several mechanisms using additive noise, in the future new mechanisms could be developed, e.g., using multiplicative noise. As we are not aware of such systems, we did not evaluate our method against theoretical implementations, and cannot make claims about the method's performance against them. We can only state that the general approach (training a machine learning classifier on query answers and using an evolutionary search to explore the space of queries) can still be applied out-of-the-box, regardless of the inner workings of the QBS.

4.9 Related work

Attacks have long been used to evaluate and improve query-based systems. Difference attacks against counting queries were studied by Denning et al. [97], as well as solutions based on randomisation [96], leading to the development of averaging attacks. The seminal linear reconstruction attack by Dinur and Nissim [101] allows to retrieve the records of a dataset using (many) counting queries, and gives a lower bound on the noise added by query-based systems to prevent it. Further work by Dwork and Yekhanin [110] showed that this attack could be performed efficiently with a smaller number of queries, and that the impossibility result extends to curators that aim to answer a small subset of queries accurately (while maintaining average noise level satisfying the lower bound). Diffix [123], developed by Aircloak, was improved through two bounty challenges [32, 33], where researchers were invited to develop attacks against it. Researchers showed that Diffix was vulnerable to reconstruction attacks using direct [73] and quasi-identifiers [72], as well as membership inference attacks [265]. Gadotti et al. proposed an attack specifically exploiting the noise structure of Diffix [129].

Query auditing [225] is a line of research which aims to detect whether queries can be answered without disclosing private information. Auditing can be performed in the offline setup, where m queries are audited together before being answered, and in the online setup, where each query is audited individually before being answered without knowledge of subsequent queries.

While the latter is known to be challenging, even the former can be difficult: for instance, query auditing is co-NPHard for counting queries [182]. Research on query auditing has been restricted to systems that answer queries exactly ($R(d,q) = T(d,q)$). QuerySnout can be seen as an automated auditor applicable to more general QBSs.

Differential privacy [107] is a formal guarantee of privacy, which was proposed as a solution to the reconstruction attack of Dinur and Nissim. Researchers have proposed query-based systems guaranteeing differential privacy, such as PINQ [213], PriPearl [177], and Flex [172]. For the counting queries we consider, all three systems behave similarly to the DPLaplace QBS we attack in this work. More complex mechanisms guaranteeing differential privacy have also been developed, such as query release through adaptive projection [108, 24] and the matrix mechanism [196, 197].

Shadow modelling. Our method uses a technique similar to shadow modelling (see Sec. 5.3 for an overview) to infer the rule G that combines answers to queries. Shadow modelling has been used extensively for property inference [23] and membership inference [302] attacks against machine learning algorithms. Pyrgelis et al. have used it to attack aggregated location data [264], and Stadler et al. have used it to develop membership inference attacks against synthetic data [312]. Machine learning-based attacks have also been used by Bichsel et al. to detect violations of DP [35].

Evolutionary algorithms have been applied to adversarial attacks against machine learning models [236, 316, 84, 16]. They are especially suitable under black-box attack scenarios where the attacker does not have access to the gradients or only hard (discrete) labels are available from the model. To the best of our knowledge, our work is the first to use evolutionary algorithms for attacks against QBSs.

4.10 Conclusion

In this work, we propose QuerySnout, the first approach to automatically discover attribute inference attacks against query-based systems. QuerySnout discovers a multiset of queries and

a rule to combine them. We learn the rule by training a machine learning classifier on answers from auxiliary QBSs protecting datasets sampled from the auxiliary knowledge available to the attacker. We use an evolutionary algorithm to find an optimal multiset of queries by iteratively improving a population of solutions using a mutation operator specifically tailored for this task. We show QuerySnout to find attacks against two deterministic real-world mechanisms, Diffix and TableBuilder, and a non-deterministic system (SimpleQBS). Across systems and datasets, the attacks found equate and often outperform previous known manual attacks. Finally, we show how QuerySnout can be extended to QBSs that require a budget and show it to approximately match the optimal accuracy against systems implementing the Laplace mechanism for large values of ε. We also discuss extensions of our attack to other attack models (e.g., membership inference) and query syntax.

Taken together, our results show how QuerySnout can be used to automatically find powerful attacks against QBSes and evaluate the privacy protection they offer. We believe that automated attack discovery procedures such as QuerySnout will help detect issues before systems are deployed, helping to patch the systems and mitigate risks, and ultimately help design query-based systems offering a high level of protection in practice.

Chapter 5

Background on machine learning privacy

This chapter presents a background on machine learning privacy. We motivate the need to evaluate privacy in ML models, describe a unified threat model for inference attacks, describe shadow modelling which is the main inference attack technique, and finally present a review of inference attacks, highlighting two open questions which we study in the next two chapters of this thesis.

The review presented in this chapter expands on the following, currently non-peer reviewed, manuscript: Gadotti, A., Rocher, L., Houssiau, F., Cretu, A.-M., and de Montjoye, Y.-A. (2023). "Anonymization: The imperfect science of using data while preserving privacy".*

The manuscript reviews the literature on privacy attacks and anonymisation techniques applied to individual-level and aggregate data. In Sec. 5.3 of this chapter, I expand on the part of the manuscript containing a literature review on privacy attacks and defences against machine learning models. I am the main contributor to this part.

5.1 Introduction

Machine learning (ML) algorithms are increasingly adopted by organisations to automate tasks and to inform or improve decision making. Examples of ML applications include dialogue

systems [247], language translation [27], voice recognition, content moderation [270], medical diagnosis [267], and drug discovery [251]. As models are being shared widely, e.g., through compiled code, or through APIs, their release raises concerns about the potential leakage of sensitive information about their training datasets.

From a privacy perspective, the datasets used to train ML models are often personal data, as they typically consist of records that encode information about individuals, such as demographics, behavioural traces, text messages, or images. A trained model may or may not be considered anonymous, depending on whether it can be linked to individuals who contributed their data to their training dataset [337]. Anonymised (or de-identified) data can indeed be shared or stored to third parties without users' consent, as it typically falls outside the scope of data protection laws such as the EU's GDPR [2] and the CCPA [3].

From a confidentiality perspective, the datasets used to train ML models are also often proprietary data. Characteristics of the dataset may be considered a trade secret or a business advantage that must not be revealed to competitors [23]. Examples include the percentage of clients in a certain age group or income bracket. In general, the expectation is that a model should learn nothing more than its intended purpose. A classification model's purpose is to estimate the probability $\Pr(Y|X = x)$ of each class given as input a training record x. Thus, a model that ends up encoding detailed information about the distribution $\Pr(X)$ might lead to unexpected leakages.

To our knowledge, it is not feasible to characterise through manual analyses what a model "remembers" (encodes) about its training dataset and under what conditions the encoded information can be extracted. A trained ML model can be seen as an aggregate of its training dataset, obtained as the result of a learning procedure, usually mini-batch gradient descent minimising a loss function with well-chosen hyper-parameters. However, we argue that models are "black-box" aggregates, due to the complex, stochastic, and iterative nature of the learning process. By "black-box", we here mean that the relationship between a model and the dataset it was trained on is opaque.[1] In most cases, including neural networks, there is not even a closed-

[1]The term "black-box" model is usually used to convey that the process by which models make decisions is opaque. Here, we use it to convey that what it encodes about its training dataset is opaque.

form expression of a trained model's parameters as a function of the dataset. Worse, when models are trained to minimise a non-convex loss function, the randomness in the training only leads to one of many local minima. Our intuition is that models "compress" the information in the dataset, keeping some information while discarding other, especially when the dataset is large and the model is small-scale, e.g., a small neural network. This is in stark contrast with aggregate statistics which can be more easily analysed. To illustrate this point, we consider a simple data query as an example of a "white-box" aggregator. Consider a database with salaries and other personal details of Imperial College London staff. Assume an adversary can query the database and knows that Alice is the only person hired in the Department of Computing (DoC) in 2022. The adversary can ask the query: "What is the average salary of DoC employees hired in 2022?". Owing to the clear relationship between the dataset and the query, we can formally conclude that when the database returns the answer as is (without adding noise), the answer is Alice's salary.

The main approach to analyse the relationship between a model and the dataset it was trained on has been to empirically evaluate the resistance of models to well-defined inference attacks. These attacks fall broadly under two categories, *privacy attacks* and *property inference attacks*. In the remaining of this chapter, we first present in Sec. 5.2 a unified threat model describing a broad range of inference attacks. Second, we review in Sec. 5.3 the literature on privacy and property inference attacks and defences against them. In particular, we describe the shadow modelling technique, the main inference attack technique against machine learning models. Finally, we present two open questions which we address in the next two chapters.

5.2 Inference attack model

In this section, we describe a general attack model – a set of assumptions on an adversary's identity, goal, and capabilities – that covers a broad range of inference attacks against machine learning models, including the ones presented in Chapters 6 and 7.

5.2.1 Setup

We consider a machine learning model $\mathcal{M}_T$ trained on a private, *target dataset* of m samples: $D_T = \{z^1, \ldots, z^m\}$. We assume the dataset to be sampled i.i.d. from an underlying distribution $\mathcal{D}$, i.e., $z^i \sim \mathcal{D}, i = 1, \ldots, m$. We here focus on classification models (also called classifiers) and assume that each sample $z = (x, y)$ consists of n variables, grouped into a record x of $n-1$ variables (e.g., an image or a list of demographic features) and a label (the output variable) from a finite set $y \in \{1, \ldots, L\}$, where L denotes the number of possible labels. The goal of the classifier is to infer the correct label y of records x sampled from $\mathcal{D}$. Towards this goal, the model computes for every sample x a probability vector $\mathcal{M}_T(x) = (\hat{y}^1, \ldots, \hat{y}^L)$, such that $\hat{y}^l \geq 0$ for $l = 1, \ldots, L$ and $\sum_{l=1}^{L} \hat{y}^l = 1$. Ideally, the l-th element $\mathcal{M}_T(x)_l$ should estimate the probability that the record belongs to the l-th class.

5.2.2 Adversary goal

The goal of the adversary is to *infer* a piece of private information about the training dataset, referred to as the *secret*, from the *target model* $\mathcal{M}_T$. Inference attacks can be categorised into privacy attacks and property inference attacks, depending on whether the secret relates to an individual or to the dataset as a whole.

In a **privacy attack**, the secret relates to an individual record in the dataset. Privacy attacks are further categorised into:

- **Membership inference attacks**, where the secret is whether a target record (x, y) belongs to the training datset D_T. Membership can be sensitive in situations where the model is trained for a sensitive task, e.g., on a dataset of individuals having a medical condition.
- **Attribute inference attacks**, where the secret is the value of one sensitive attribute x_1 for a target record $(x, y) = ((x_1, \ldots, x_n), y) \in D_T$.
- **Reconstruction attacks**, where the secret is an entire training record x, e.g., an image.

In a **property inference attack**, the secret is a summary statistic of the dataset, e.g., the propertion of females in the dataset or the average node degree of a graph.

5.2.3 Adversary access to the model

We consider an adversary having some degree of access to the *target model* $\mathcal{M}_T$. More specifically, we assume at the very least that the adversary can query the model on valid samples x to retrieve the output probabilities of the model. The adversary having only query access to the model is typically known as *black-box adversary*. Most commonly, this adversary is assumed to be able to retrieve the entire probability vector $(\hat{y}^1, \ldots, \hat{y}^L)$. Some works have also considered weaker adversaries that e.g., have access to only the k highest probabilities or the k most likely classes [283, 68].

In many cases, models are made available as a *white box*, e.g., when they are publicly released for scientific purposes or deployed on user devices. White-box access to the model allows an adversary to additionally inspect the model architecture and internal parameters (also called *weights*). In the case of neural networks, knowledge of the loss function allows the adversary to compute gradients on any inputs. It is reasonable to assume the adversary knows the loss when the target model is a classifier, as classifiers are commonly trained with standard loss functions (i.e., the cross-entropy loss). Importantly, while the number of queries to black-box models may be restricted in case of suspicion of malicious behaviour or due to limited resources, an unlimited number of queries can be asked to models released as a white box, since the weights describe the functioning of the model completely.

5.2.4 Adversary knowledge

It is important to clarify what an adversary knows and what they do not know (i.e., their uncertainty) about the target model $\mathcal{M}_T$ and dataset D_T. In general, weak adversaries have higher uncertainty and are considered more realistic. Strong adversaries can, in some cases,

also be argued to be realistic or useful, e.g., a defence developed to protect against a strong adversary will also protect against weaker ones.

In the context of machine learning, the adversary uncertainty typically relates to the training algorithm and to the private dataset.

Uncertainty about the training algorithm. A machine learning model is the result of a training algorithm $\mathcal{T}$ executed on the training dataset. The training algorithm implements an optimisation procedure whose goal is to fit the internal parameters of the model in order to minimise a differentiable loss function. The loss function measures how well the model is able to classify samples of the distribution, with lower values indicating better performance. The training algorithm encompasses the choice of network architecture, loss function, hyperparameters such as learning rate and the batch size (for mini-batch gradient descent). For non-deterministic optimisation procedures such as those minimising a non-convex loss function (which is standard in neural networks), the training algorithm additionally encompasses the randomness of model initialisation, the stochasticity of gradient descent (ordering of mini-batches), and the randomness of dropout regularisation [310].

Many of the empirical evaluations in the literature (e.g., [302, 68, 365, 52]) assume that the adversary has full knowledge of the training algorithm $\mathcal{T}$, with the exception of the initialisation seed (that fixes the randomness of the algorithm)[2]. This assumption allows the adversary to train so-called *shadow models*, i.e., models similar to the target model on datasets of their choice (see Sec 5.3 for details). Some works however relax such assumptions, e.g., to make the attack applicable to a setup such as MLaaS where the user cannot specify the model architecture or hyperparameters [302], to understand the limits of attacks [283, 52], or simply to develop simpler methods that do not require shadow models [354].

Uncertainty about the private dataset. In most cases, the adversary can be assumed to have *some* information about the dataset. For instance, the attribute inference adversary aiming to infer the first attribute x_1 typically knows the rest of the record $(x_2, \ldots, x_n)$ and the label y, and is certain that the record was used to train the model. In most cases, the uncertainty

[2]This assumption is rarely made explicit in the literature.

of the attacker is modelled as access to an auxiliary dataset with the same distribution $\mathcal{D}$, and some works even assume access to part of the dataset [215, 229, 29]. Property inference attacks assume the adversary can sample from a generic distribution $\mathcal{D}'$ and that the training dataset D_T is sampled from $\mathcal{D}'$ so as to respect a *secret* proportion along a given attribute. The adversary goal is to determine which one of several possible choices for the proportion D_T was sampled from. Thus, by design, the adversary cannot determine the proportion by merely computing its expected value under $\mathcal{D}'$.

To simplify the analysis, a common approach is to restrict the set of values the *secret* can take, from its full range to a few possibilities. For instance, most property inference attacks try to distinguish between two specific proportions, e.g., 23% or 37% of older faces [131], even though theoretically the proportion can range between 0% and 100%. The threat model thus assumes a stronger adversary that could be defended under scenarios where the adversary has a good guess on the value of the secret.

Rigourous and well-defined assumptions on the adversary uncertainty about the dataset allow to reason about baselines that do not exploit access to the model. This allows to quantify the leakage from the model by comparing it with what can be already deduced from the adversary knowledge alone, and is at the core of our contribution in Chapter 6.

5.3 Literature review on inference attacks

5.3.1 Privacy attacks

Privacy attacks against ML models are a very active area of research, mostly focusing on membership inference attacks [302, 354, 283, 229, 193, 205, 68, 52], but including also attribute inference attacks [127, 215, 167] and reconstruction attacks [29, 54, 282]. These attacks have studied adversaries with different auxiliary information and levels of access to the model: black-box query access—giving the adversary access to model predictions, as is typical in the context of machine learning as a service (MLaaS)—and white-box access—also giving the adversary

access to the model's internal parameters, as is typical in the context of on-device deployment.

Membership inference attacks. Shokri et al. [302] proposed the first membership inference attack (MIA) against ML models. Their seminal work demonstrated how models trained via popular MLaaS platforms (developed by Google and Amazon) without any defences can leak with high accuracy the membership of records. Their attack quantifies the membership leakage of classification models via their black-box confidence scores, where each score is a proxy for the probability that the record belongs to the corresponding class. Intuitively, models tend to be more confident on their training records compared to unseen records. Their methodology is based on training the same model on similar datasets with and without the record of interest before training a meta-classifier to predict whether the record was part of the training dataset or not. Known as the *shadow modelling* technique, this generic approach was first developed by Ateniese et al. [23] (although not by this name) and is the main building block of most inference attacks against ML models. This attack was later extended to other setups, including models that only output the prediction label instead of detailed confidence scores for all the classes [68] and models released as a white-box [229, 193].

The intuition behind the shadow modelling technique is that inferring a secret about the target model's training dataset, such as the membership of a target record x, involves learning a distinguishing boundary between models trained on datasets having different values of the secret. To achieve this, the adversary uses their knowledge about the training algorithm $\mathcal{T}$ to train one or more *shadow models* $\mathcal{M}_1, \ldots, \mathcal{M}_K$ on *shadow datasets* $D_1, \ldots, D_K$ sampled from the adversary knowledge. The adversary then extracts from each model $\mathcal{M}_k$ a set of features F_k deemed useful for the inference task, such as confidence scores on well-chosen records [302] or weights [131]. The features are labeled with the value of the secret S_k in the corresponding shadow dataset D_k. Finally, a meta-classifier $\mathcal{A}$ is trained to perform the inference task such that $\mathcal{A}(F_k) \approx S_k$. For attacks such as membership inference targeting a secret about one record, it is not necessary to train shadow models such that the targeted record is a member of the training dataset in half of them. Indeed, the meta-classifier can be trained to distinguish between confidence scores computed on training and unseen records extracted from only one shadow model [283], albeit at a cost in accuracy [52].

The accuracy of MIAs is generally impacted by the size of the training dataset, the level of overfitting (i.e. the gap between the model accuracy on training and on unseen records from the same distribution), the model architecture, and the number of classes. For instance, models of the same architecture are generally less vulnerable when they are less overfitted [302]. However, for a similar level of overfitting, some architectures leak more membership information than others, with recent results suggesting that more accurate models may be more vulnerable [52]. Interestingly, in some cases even simple classifiers such as logistic regression or decision trees may leak membership of their records, particularly on high-dimensional datasets with a large number of classes [332]. The vulnerability generally decreases with the training dataset size and increases with the number of classes [302]. Finally, the accuracy of the adversary's auxiliary information on the target record also impacts MIA performance (the noisier the record, the lower the effectiveness of the attack) [332]. In general, expressive models—such as overparameterised neural networks—may memorise the presence of at least a few records, particularly rare or atypical ones. Recent theoretical work suggests some amount of memorisation may in fact be needed to achieve optimal model utility on long-tailed distributions [118]. Empirically, the risk can indeed be higher for records belonging to small subpopulations [188] or outlier records [205]. The common metric used to quantify MIA vulnerability—the average success rate—may not directly reveal such cases, as an attack may perform poorly on the average record but still very confidently infer the membership of some records. Recent work has argued that MIAs should be considered a risk even when they confidently target only a few records [205, 52].

Black-box MIAs have received great attention because they are applicable to both black-box and white-box attack settings and they can be used as a simple test of the privacy offered by a model. Indeed, when successful, they can point to leakages in the training pipeline. The standard and state-of-the-art attack methodology however relies on the rather strong assumption that the adversary has access to many records of the underlying data distribution (used to train the shadow models) and has knowledge of the precise training algorithm and hyperparameters used. Although these assumptions reflect an auditor more than a realistic adversary, leakages detected in this way should not be dismissed. In practice, when releasing a model it is important to evaluate on a case-by-case basis whether the hypotheses underlying the attacks are met. The

few works that examined more realistic adversaries suggest that the performance of MIAs can be impacted when relaxing some of the aforementioned assumptions [302, 354, 283, 52].

Attribute inference attacks were first studied by Fredrikson et al. [127] in the black-box setting, and later improved by Mehnaz et al. [215] and extended to the white-box setting [126, 167]. The study by Fredrikson et al. was grounded in a real-world use case, asking whether a model to predict a patient's dosage of the warfarin drug can be exploited to learn a patient's genetic marker. Their attack returns the most likely value for the sensitive attribute. The probability of each candidate value is estimated using the one-way marginals of the dataset (which the adversary is assumed to know) multiplied by a quantity estimated from the model's confusion matrix[3]. Privacy researcher Frank McSherry called into question the conclusions of this study, arguing that since genetic markers are correlated with the dosage and the adversary's prior knowledge includes the dosage, it is unclear whether the model leaks more than what can be inferred from the prior knowledge [212]. A similar argument was put forward recently by Jayaraman and Evans [167], who evaluated the performance of various attribute inference attacks against baseline approaches that perform imputation based on prior knowledge available on the distribution. Their results suggest that current black-box attacks may not be able to extract more information about the target than the imputation baseline, and that white-box attacks might. They also highlight the need to distinguish, when evaluating attacks, between inferences made possible through access to a model and population-level inferences that are possible even without access to the model.

Reconstruction attacks aim to recover one or more records of the training dataset given access to the model. Intuitively, generative models (a class of machine learning models that generate data similar in distribution to the training dataset) are more directly vulnerable to reconstruction attacks than classification models, as their goal is to produce outputs similar to the training dataset. Language models for text generation have been shown to output sensitive text sequences from their training datasets, including personally identifiable information such as names or social security numbers [53, 54]. In the context of classification models, reconstruction

[3]This is the probability that the true label of a record is y conditionally on the probability that the model predicts the label $\hat{y}$ observed for the candidate record.

attacks have so far only been demonstrated in constrained setups. In 2015, Fredrikson et al. [126] proposed techniques for extracting a representative sample of a class. Balle et al. [29] later showed that a "worst-case" adversary—one that has complete knowledge of the dataset except for one record and white-box access to the model—can reconstruct the unknown record from the model parameters. While such a strong adversary model may be useful to measure worst-case information leakage, the risk measured does not translate to more practical settings.

Defending against privacy attacks on ML models is presently a very active area of study. Training models with differential privacy guarantees [107] is currently seen as the most principled way to prevent privacy attacks, although how to achieve comparable utility (model accuracy) to non-private models (i.e., models produced without privacy guarantees) remains an open problem. Early work [61] extended the output perturbation approach of Dwork et al. [107] to various classification models. The most popular approach today is to introduce noise during the training, through objective perturbation by adding noise to the loss function, e.g. in logistic regression [61, 62], or through gradient perturbation [308, 8]. Known as DP-SGD, the latter algorithm is implemented in the main deep learning libraries, including PyTorch through the Opacus API [357] and Tensorflow [7]. Other techniques include the private aggregation of teacher ensembles (PATE) [254] that also provides differential privacy guarantees.

The main limitation of differentially private ML is that using low values for the privacy loss ε that give strong privacy guarantees also results in much lower utility compared to the non-private models [166, 26]. Indeed, efficiently adding noise tailored to each model is difficult [253]. Training a model involves selecting hyperparameters, which is hard to do with differential privacy as it requires multiple passes over the dataset, each using a fraction of the budget. Thus, properly integrating differential privacy in the model development process involves privacy-preserving hyperparameter tuning [255]. For domains where a lot of data is easily available, using a larger private training dataset, a similar public dataset, or robust handcrafted features can improve utility [328]. Some architecture choices that have become dominant based on their strong performance in the non-private setup may achieve sub-optimal utility when trained using DP-SGD, with recent work arguing for the development of privacy-aware architectures [256]. For instance, Papernot et al. [256] show that models using a bounded non-linearity function

such as the hyperbolic tangent can achieve much better utility than the standard ReLU.

Fortunately, empirical evidence suggests that even large values of ε might be sufficient to prevent some attacks [193, 53]. This potentially points to a gap between the theoretical privacy afforded by a mechanism against the strongest possible adversary assumed by differential privacy and its robustness to weaker, practical adversaries. This gap can be explained by formally analysing the privacy loss under weaker adversary assumptions, but this is considered difficult in non-convex models [230]. A practical alternative is to develop auditing approaches [163, 230, 359, 206], where well-defined adversaries (that are weaker than the "worst-case" one implicitly assumed by differential privacy) are deployed against a mechanism. The attack performance of these weaker adversaries is used to empirically derive a lower bound of the privacy loss. Put informally, a model that leaks this amount of individual-level information must have been trained with a larger ε than the estimated lower bound.

5.3.2 Property inference attacks

Property inference attacks (PIA) study the leakage of macro properties of the training dataset. Their primary motivation is to infer the percentage of training samples along a sensitive attribute, with applications such as auditing (verifying that a model is trained on representative data) and detecting leakages of trade secrets, where a model leaks information unrelated to its design purpose. Ateniese et al. [23] introduced property inference attacks (PIAs), although by another name, and the meta-classifier technique now known as shadow modelling. The study applied PIAs to Hidden Markov Models and Support Vector Machines and showed that it is possible to distinguish the dialect on which a speech recognition models was trained on. Ganju et al. [131] showed that extending this attack to multilayer perceptrons (MLP) is more challenging due to the permutation equivalence property of MLPs, where neurons of the hidden layers can be arbitrarily permuted without changing the underlying function. Thus, a meta-classifier trained on the internal parameters of the model needs to learn a more difficult task: (1) the relationships between parameters and the target property and (2) that this relationship is invariant to neuron permutations. The authors propose two approaches to tackle this challenge:

weight-based neuron sorting and set-based meta-classifiers. They show these approaches can outperform the naïve one that trains a classifier on the model parameters in the "given" order. Zhang et al. [365] proposed the first *black-box* property inference attack, making use of the model's output probabilities computed on a dataset of real records from the same distribution. Finally, while most of the works have focused on classification models, Zhou et al. [368] recently extended PIAs to Generative Adversarial Networks (GANs) and showed that they can be used to enhance MIAs against GANs by calibrating a sample's membership score using the inferred property.

Recently, Suri and Evans [318] proposed a unified threat model generalising assumptions and goals of PIAs from prior work. Concretely, the adversary aims to distinguish between two worlds, each capturing a different distribution of the training dataset. The two distributions are obtained by applying different transformations $\mathcal{G}_1$ and $\mathcal{G}_2$ to a distribution $\mathcal{D}'$, e.g., by sampling records with probabilites α_1 or α_2. The adversary knows $\mathcal{G}_1$ and $\mathcal{G}_2$ as well as $\mathcal{D}'$. The authors furthermore extend the binary distinguishing game to a regression task aiming to predict the unknown probability used to sample training dataset and show that property regression performs better than binary classification, probably because it is trained on more diverse data.

There are currently no known effective defences against PIAs. Differential privacy applied to the k-means clustering algorithm was shown not to mitigate the attack [23]. This is expected as differential privacy is not designed to protect population-level information. Removing the attribute whose proportion the adversary aims to infer from the data can reduce the leakage but not completely [365].

5.4 Open questions

We identify two open questions that we study in this thesis.

The first open question is characterising what a model "remembers" about the distribution of its training dataset D. Indeed, from a machine learning perspective, little is known today about

what a model memorises about its training distribution beyond the expected $\Pr(Y|X = x)$. This is a non-trivial question. On the one hand, the ideal model should learn nothing more than the intended probability of the output label given an input sample $\mathcal{M}(x) \simeq \Pr(Y|X = x)$. However, models are trained to minimise a loss function computed over a specific finite dataset of samples from the underlying dataset distribution $\Pr(X, Y)$ [140]. The dataset distribution $\Pr(X, Y)$ and, by extension, $\Pr(X)$, may very well end up influencing the final model in practice.

Property inference attacks have studied whether models leak the percentage of records having specific values along a targeted attribute [131, 365, 318]. These studies commonly assume that the adversary has the ability to sample real records from a similar distribution to run the attack. This assumption is restrictive and inapplicable to settings where similar data is not widely available, e.g., medical data. An important question is whether an adversary can infer information about the dataset distribution *without access to real records*.

In particular, can an adversary learn the correlations between input variables of the training dataset? Correlations are important to study, for two reasons. First, privacy attacks require defining a prior about the dataset. An adversary that *does not* have access to real records can use a probabilistic model as the prior. Since variables in the real world are very often statistically dependent, the prior needs to model the joint variable distribution, and correlations are one of the minimal summary statistics the prior should capture. Second, correlations can be building blocks for other attacks, in particular enabling weaker adversaries. For instance, if an adversary learns the correlation between Alice's public attributes and a private, sensitive attribute from access to a model trained to perform a task on these attributes, the adversary may use this knowledge to improve their prior about Alice's sensitive attribute. Since correlations are population-level characteristics, they are not protected by differential privacy guarantees [107]. Yet, if an adversary learns information about the specific dataset a model was trained on that *is not public knowledge nor easily retrievable information*, and uses it to learn something new about an individual whose data was used to train the model, this may lead to a privacy violation [85, 77]. [4]

[4]Cormode et al. [77] made a similar argument about aggregate statistics (e.g., population counts and contingency tables). We here argue that it also applies to models since a model can be seen as an "aggregate" of the dataset it was trained on.

The second open question is evaluating the impact of deep neural network (DNN) symmetries on white-box MIAs. More specifically, we focus on DNNs such as multilayer perceptrons (MLP) and convolutional neural networks. DNNs exhibit certain symmetries [151, 21], meaning that the same model can have many different weight representations. The existence of symmetries may lead to a lack of correspondence between shadow models and the target model, hindering the meta-classifier's task.

An open and important problem is analysing the impact of symmetries on white-box MIAs. Indeed, white-box MIAs do not, in general, perform much better than black-box MIAs [281, 229, 193]. White-box property inference attacks against MLPs have been shown to be affected by symmetries [131]. Our hypothesis is that if white-box MIAs are also affected by symmetries, addressing the issue can lead to stronger attacks and thus allow for a tighter evaluation of privacy risks.

Chapter 6

Correlation inference attacks against machine learning models

The work presented in this chapter is based on the following, currently non peer-reviewed, manuscript: Cretu, A.-M., Guepin, F.*, and de Montjoye, Y.-A. (2022). "Correlation Inference Attacks against Machine Learning Models". A version of this manuscript is available at* `https://arxiv.org/abs/2112.08806`.

Florent Guépin and I are joint first authors on this paper. We collaborated in the design of the method and of the experiments, in the writing of the code, and in the writing of the paper. My main contribution is in conceptualising the attack, designing the methods and the experiments, writing part of the code especially the synthetic data experiments, MLP evaluation and scaling of the evaluation, generating the figures, simplifying, refining and improving the methods, and writing the paper.

6.1 Introduction

In this chapter, we study a new type of leakage in machine learning (ML) models: the leakage of correlations between input variables of a tabular training dataset. Information leakage

by ML models has, so far, been mostly studied through the lens of *individual records*. This includes membership inference attacks [302, 283, 354, 229, 68, 52] aiming to infer if a record was used to train a model – and to a lesser extent, attribute inference [127, 126, 215] and reconstruction [282, 29] attacks – aiming to reconstruct respectively, partially and entirely, a record. While potentially very invasive from a privacy perspective, attacks against individual-level records commonly rely on powerful attackers, with partial [229] or near-complete [29] knowledge of the dataset, or with access to a dataset sampled from the same distribution. As a result, they may not reflect the information leakages achievable by attackers lacking access to a similar dataset.

We argue that evaluating the information leakage of a model requires quantifying its susceptibility to attacks aiming to extract information about both *individual records* and *summary statistics*. Summary statistics can take different forms such as k-way marginals and correlations or macro properties of the dataset (see Sec. 4.9 for a discussion of property inference attacks). Sensitive k-way marginals could, for example, reveal the profile of people registered to an alcohol rehabilitation clinic from an admission prioritisation model trained on their data. Combined with one-way marginals, correlations would, in the first example above, also allow for much more detailed profiling of the patients registered with the admission clinic. Worse, even models trained on representative data could leak information unrelated to the task they have been trained for. For instance, a scoring model for depression could reveal that patients living in the inner city are more likely to have used illegal substances. Finally, summary statistics can be building blocks for other attacks including on individual records.

Contributions. We argue that understanding whether ML models leak the correlations between their input variables is an open and important question (to avoid repetition, we point the reader to Sec. 5.4 for the motivation). We propose what is, to our knowledge, the first *correlation inference attack against ML models*. We propose two different attacks. First, a *model-less* attack that uses knowledge of the correlations between the input variables X and the output variable Y, which we call the *correlation constraints*, to predict the correlation between two input variables $\rho(X_1, X_2)$. We argue that the correlation constraints are typically known to or inferrable by the attacker. Our attack exploits the spherical parametrisation of

correlation matrices [261, 241], imposing bounds on the correlation coefficients, to make an informed guess. This first approach also acts as a strong baseline, allowing us to correctly quantify the leakage caused by the machine learning model. We then propose a second, *model-based* attack which exploits access to an ML model trained on the dataset of interest, using shadow models [23, 302]. However, unlike prior works [23, 131, 365, 52, 318], our shadow models are only trained on synthetic data, greatly reducing the strength of the adversary compared to the literature. Our synthetic data generation approach combines Gaussian copula-based generative modelling [307] with carefully adapted procedures for sampling correlation matrices under constraints. Our choice of the Gaussian copula prior is motivated, first, by its past success in modelling tabular data in various applications [66, 329, 278]. Second, it is a generative model parameterised by one-way marginals and a covariance matrix [307], allowing us to generate training datasets for the shadow models which match the adversary's knowledge about the target dataset, while spanning the entire range of values attainable by the unknown correlations.

Second, we study the performance of our model-based attack on synthetic data under three different attack scenarios. We show it to consistently strongly outperform the model-less attack, proving that ML models such as Logistic Regressions (LR) and Multilayer Perceptrons (MLP) leak information about the correlations between their input variables X. We show how mitigations such as limiting the number of queries that can be made to the model or their precision fail to prevent the attack. We also show that ensuring that the model is differentially private (DP) [107] does not prevent our attack.

Third, we study the performance of our attack on three real-world datasets. Our results confirm that ML models leak information about the correlations between their input variables X beyond what would be available using the correlation constraints alone.

Finally, we show how summary statistics can be building blocks for other attacks by proposing a novel attribute inference attack exploiting the correlations extracted by our model (CI-AIA). We show CI-AIA to significantly outperform previous attacks on real-world datasets.

6.2 Correlation inference attack

6.2.1 Attack model

We consider a machine learning model $\mathcal{M}_T$ trained on a *target dataset* D_T consisting of m records $z^1, \ldots, z^m$. Each z^i is a sample of $n-1$ input variables $X_1, \ldots, X_{n-1}$ and one output variable Y. We assume variables to be continuous and real-valued and denote by $F_i(x) = \Pr(X_i \leq x)$ the one-way marginal of the i-th input variable and by $F_n(y)$ the one-way marginal of the output variable.

For simplicity, we consider $\mathcal{M}_T$ to be a binary classifier whose goal is to infer if $Y > c$, where c is a task-specific threshold such as the average or the median value. Our attack can however be easily generalised to multi-class classification and regression tasks.

Our adversary aims to infer the Pearson correlation coefficient $\rho(X_1, X_2)$, between two input variables X_1 and X_2 given access to the machine learning model $\mathcal{M}_T$. Also called *linear correlation*, it measures the strength of the linear association between two variables and is widely used by researchers and data scientists across many domains [343, 367]. We provide its formal definition below, and call $\rho(X_1, X_2)$ the *target correlation* and X_1 and X_2 the *target variables*.

Definition 6.1. *Denote by $E(X)$ the expectation of a random variable X. The Pearson correlation coefficient between two random variables X_1 and X_2 with finite variance $Var(X_i) = Cov(X_i, X_i) < \infty, i = 1, 2$ is defined as:*

$$\rho(X_1, X_2) = \frac{Cov(X_1, X_2)}{\sqrt{Cov(X_1, X_1)}\sqrt{Cov(X_2, X_2)}} \tag{6.1}$$

where $Cov(X_1, X_2)$ denotes the covariance between the two variables and is defined as:

$$Cov(X_1, X_2) = E[(X_1 - E(X_1))(X_2 - E(X_2))] \tag{6.2}$$

6.2.2 Adversary knowledge on data distribution

We make two assumptions on the adversary knowledge on the data distribution.

Knowledge of one-way marginals. First, we assume the adversary to have knowledge of the one-way marginals of all the variables $F_i(x), i = 1, \ldots, n-1$ and $F_n(y)$. These are indeed typically released, e.g., in research papers or to comply with legal requirements such as the European Union AI Act [75] and their knowledge is a common assumption in the literature [127, 126, 365].

Knowledge of some correlations. Second, we assume the adversary to have access to the linear correlations between some of the variables. More specifically, we define the *partial knowledge* of the adversary as a set P of pairs such that for every $(i, j) \in P$ the adversary has the knowledge of the ground-truth values $\rho(X_i, X_j)$ (with $X_n = Y$). As $\rho(X_i, X_j) = \rho(X_j, X_i)$, by symmetry we assume that P consists of ordered pairs, i.e., if $(i, j) \in P$, then necessarily $i < j$.

We consider three different attack scenarios:

(S1) The adversary knows the correlations between the target variables and the output variable, namely $\rho(X_1, Y)$ and $\rho(X_2, Y)$, i.e., $P = \{(1, n), (2, n)\}$, but not $\rho(X_i, X_j)$ for $1 \leq i < j \leq n-1$ nor $\rho(X_i, Y)$ for $i \geq 3$.

(S2) The adversary knows the correlations between all the input variables and the output variable: $(\rho(X_i, Y))_{i=1,\ldots,n-1}$, i.e., $P = \{(i, n) \ : \ i = 1, \ldots, n-1\}$, but the adversary does not know $\rho(X_i, X_j)$ for $1 \leq i < j \leq n-1$.

(S3) The adversary knows all the correlations between the variables, except for the target correlation $\rho(X_1, X_2)$. More specifically, the adversary knows $\rho(X_i, X_j), \forall 1 \leq i < j \leq n$ such that $(i, j) \neq (1, 2)$, i.e., $P = \{(i, j) \ : \ 1 \leq i < j \leq n\} \setminus \{(1, 2)\}$.

Note that the three scenarios are identical for $n = 3$ variables.

We will henceforth call the known coefficients the *correlation constraints*.

S2 is the default scenario considered in this work. This choice is motivated by the fact that the relationship between the input and output variables – encoded through $\rho(X_i, Y)$ – is unlikely to be considered a secret, since the model's goal is to infer Y given X. We show in Sec. 4.8 how these correlation constraints can be extracted from the model.

S1 is a weak adversary, motivated by our goal to measure the *leakage from the model*, rather than the leakage derived from constraints imposed on X_1 and X_2 via their association with the output variable Y. Indeed, our analysis shows that the correlation coefficients $\rho(X_1, Y)$ and $\rho(X_2, Y)$ restrict the range of values that the target coefficient $\rho(X_1, X_2)$ can take, without the adversary even accessing the model.

S3 is the strongest, "worst-case" scenario, corresponding to an adversary having complete knowledge of the correlations, except for the target. We study this scenario in order to evaluate how much information a very strong adversary would be able to infer.

6.2.3 Adversary access to the model

We consider two different assumptions on the adversary access to the target machine learning model $\mathcal{M}_T$.

Model-less attack. Under this attack scenario, the adversary does not have access to the target model $\mathcal{M}_T$. This scenario mostly serves as a stronger than random baseline allowing measuring the adversary's uncertainty on the target correlation under their prior knowledge. It might also become an attack in situations where the model is not released, yet information about the marginals and the correlations between the input variables and the target variables is made available, e.g., as part of a research study.

Model-based attack. Under this attack scenario, we assume the adversary (1) knows the model architecture and training details, e.g., number of training epochs for a neural network, allowing them to train from scratch a similar model and (2) has black-box query access to the target model $\mathcal{M}_T$, allowing to retrieve the output probabilities for each class $\mathcal{M}_T(x)$ on inputs x.

6.2.4 Correlation inference task

We define the *correlation inference* task as a classification task by dividing the range of correlations $[-1, 1]$ into B bins of equal length. When $B = 3$, the bins are $[-1, -1/3)$, $[-1/3, 1/3)$ and $[1/3, 1]$, which we refer to as *negative*, *low* and *positive* correlations, respectively.

We opt for a classification rather than a regression task because it is easier to analyse the attack performance by comparing its accuracy with a random guess baseline (at $1/B$). Unless otherwise specified, we use $B = 3$ classification bins in our empirical evaluation, as the correlation ranges can be easily interpreted. We also discuss in Sec. 6.5.2 and Sec. 6.5.4 results of our attack for $B = 5$ bins.

6.3 Attack methodology

6.3.1 Overview

At a high level, our methodology analyses the behaviour of the target model through the confidence scores returned on specific inputs, searching for patterns allowing to infer the correlations between input variables. More specifically, we train a meta-classifier to infer the correlation $\rho(X_1, X_2)$ between two variables of interest in a dataset D based on outputs extracted from a model $\mathcal{M}$ trained on D. Training the meta-classifier requires a large number of models $\mathcal{M}$ derived from datasets D spanning a wide variety of values for the unknown correlations $\rho(X_i, X_j)$, where $(i, j) \notin P$, while matching the correlation constraints $\rho(X_i, X_j)$ for every $(i, j) \in P$. Generating such correlations, and datasets that satisfy them, is challenging.

First, to formally define the linear relationships between the n variables in the dataset, we recall in Sec. 6.3.2 the concept of *correlation matrix*. We then describe the spherical parametrisation of correlation matrices, which we use to theoretically bound the range of values attainable by $\rho(X_1, X_2)$.

Second, we translate this range into an attack, that leverages only the correlation constraints

known to the adversary, but not the access to the target model, to make a guess (*model-less attack*, Sec. 6.3.3).

Third, we develop procedures to sample correlation matrices satisfying the correlation constraints available to the adversary (Sec. 6.3.4).

Fourth, we describe how we generate synthetic shadow datasets from a correlation matrix using Gaussian copulas, and how we adapt this approach to generate datasets having arbitrary one-way marginals (Sec. 6.3.5).

Finally, we develop a *model-based attack* by training a meta-classifier on models trained on synthetic datasets satisfying the correlation constraints and spanning the entire range of values attainable by the unknown correlations (Sec. 6.3.6).

6.3.2 Spherical parametrisation of correlation matrices

Correlation matrix. Given n random variables, we denote by C their correlation matrix, i.e., the $n \times n$ real-valued matrix of Pearson correlation coefficients between all the pairs of variables. Using lowercase notation to denote matrix coefficients, the correlation matrix is defined as $c_{i,j} = \rho(X_i, X_j), \forall 1 \leq i, j \leq n$. Let $\mathcal{C}$ be the set of valid correlation matrices consisting of all $n \times n$ real-valued matrices satisfying the following properties:

(P1) All elements are valid correlations, i.e., real values between -1 and 1: $-1 \leq c_{i,j} \leq 1, \forall 1 \leq i, j \leq n$.

(P2) Diagonal entries are equal to 1, i.e., there is perfect correlation between a variable and itself: $c_{i,i} = 1, \forall 1 \leq i \leq n$.

(P3) The matrix is symmetric, i.e., the correlation between X_i and X_j is the same as the correlation between X_j and X_i: $c_{i,j} = c_{j,i}$.

(P4) The matrix is positive semi-definite: $x^T C x \geq 0, \forall x \in \mathbb{R}^n$.

The intuition at the core of our attacks is that the correlation constraints allow to derive theoretical limits on the range spanned by the other values. First, our model-less attack combines

the properties of valid correlation matrices with the correlation constraints to determine the range spanned by the target $\rho(X_1, X_2)$. Often a strict subinterval of $[-1, 1]$, this range can be used to infer $\rho(X_1, X_2)$ more accurately than a random guess. Second, our model-based attack samples shadow datasets matching the correlation constraints, and spanning all the possible values for the unknown correlations. We do this by sampling correlation matrices under constraints, then using them to generate synthetic shadow datasets.

The **spherical parametrisation** of correlation matrices [261] provides a principled and effective approach for sampling valid correlation matrices. As C is a positive semi-definite matrix, it has a Choleski decomposition $C = BB^T$, where B is lower triangular with non-negative diagonal entries: $b_{i,j} = 0, 1 \leq i < j \leq n$ and $b_{i,i} \geq 0, i = 1, \ldots, n$. As $1 = c_{i,i} = (BB^T)_{i,i} = \sum_{j=1}^{n} b_{i,j}^2 = \sum_{j=1}^{i} b_{i,j}^2$, it follows that each vector $(b_{i,1}, \ldots, b_{i,i}) \in \mathbb{R}^i$ has an $\mathbb{L}_2$ norm of 1 and that it can be expressed using spherical coordinates, i.e., $\exists \theta_{i,j} \in [0, \pi), 1 \leq i < j < n$ such that:

$$B = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ \cos\theta_{2,1} & \sin\theta_{2,1} & \ddots & \ddots & \vdots \\ \cos\theta_{3,1} & \cos\theta_{3,2}\sin\theta_{3,1} & \sin\theta_{3,2}\sin\theta_{3,1} & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \cos\theta_{n,1} & \cos\theta_{n,2}\sin\theta_{n,1} & \cos\theta_{n,3}\sin\theta_{n,2}\sin\theta_{n,1}\cdot & \cdots & \prod_{j-1}^{n-1}\sin\theta_{n,j} \end{pmatrix} \tag{6.3}$$

Note that as expected, only $\frac{n\times(n-1)}{2}$ parameters, called angles, are sufficient to characterise the set of correlation matrices.

6.3.3 Model-less attack

Given the correlation constraints $(c_{i,j})_{(i,j)\in P}$, we call *matching set* C_P the set of valid correlation matrices satisfying them: $C_P = \{C' \in \mathcal{C} \ : \ \forall (k,l) \in P \ , \ c'_{k,l} = c_{k,l}\}$. Due to the correlation constraints, the range of values R that can be taken by $c_{1,2}$ is a subinterval of $[-1, 1]$:

$$R = [\inf_{C'\in C_P} c'_{1,2}, \sup_{C'\in C_P} c'_{1,2}] := [m_1, m_2] \tag{6.4}$$

Our model-less attack outputs the majority bin over this interval. To avoid introducing biases in the evaluation, we assume the uniform prior over this interval and return the classification bin which is covered in the largest proportion by $[m_1, m_2]$.

For preciseness and reproducibility, we present the complete details below. First, the B classification bins are determined by the following intervals:

$$\begin{gathered}\left[\frac{2(b-1)-B}{B}, \frac{2b-B}{B}\right), \text{ for } b = 1, \ldots, B-1 \\ \left[\frac{2(b-1)-B}{B}, \frac{2b-B}{B}\right], \text{ for } b = B\end{gathered} \tag{6.5}$$

Denoting by $[m_1, m_2]$ the range of possible values for target correlation $\rho(X_1, X_2)$, our model-less attack predicts the majority bin over this interval, distinguishing three cases:

(C1) The interval is fully included inside a bin, i.e., $(2(b-1)-B)/B \leq m_1 \leq m_2 < (2b-B)/B$ for some $b = 1, \ldots, B$. We predict the bin b.

(C2) The interval $[m_1, m_2]$ partially covers two bins, but none entirely, i.e., $m_1 < (2b-B)/B \leq m_2 < (2(b+1)-B)/B$, for some $b = 1, \ldots, B-1$. We predict the bin b if it has higher coverage than $b+1$ (i.e., if $(2b-B)/B - m_1 > m_2 - (2(b+1)-B)/B$), and $b+1$ otherwise.

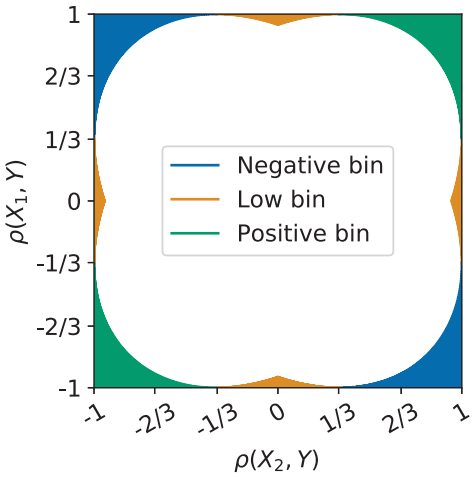
(C3) At least one bin is fully covered by the interval $[m_1, m_2]$. We predict one of the bins, sampled uniformly at random.

As an example, consider the case $n = 3$ and an adversary having knowledge of $\rho(X_1, Y)$ and $\rho(X_2, Y)$ (corresponding to $P = \{(1,2), (1,3)\}$). We assume, for simplicity, that the correlation matrix is computed over Y, X_1, X_2 in this order. Developing the computation $C = BB^T$ yields the following correlation coefficients:

$$\begin{aligned} c_{2,1} &= \cos\ \theta_{2,1} = \rho(X_1, Y) \\ c_{3,1} &= \cos\ \theta_{3,1} = \rho(X_2, Y) \\ c_{3,2} &= \cos\ \theta_{2,1} \cos\ \theta_{3,1} + \sin\ \theta_{2,1} \sin\ \theta_{3,1} \cos\ \theta_{3,2} = \rho(X_1, X_2) \end{aligned} \tag{6.6}$$

Since $\theta_{2,1}$ and $\theta_{3,1}$ are fixed, the only degree of freedom left on the unknown $\rho(X_1, X_2)$ is $\theta_{3,2}$. Any choice of $\theta_{3,2} \in [0, \pi)$ yields a valid correlation matrix. The unknown $\rho(X_1, X_2)$ can thus take any value between $\cos\,\theta_{2,1} \cos\,\theta_{3,1} - \sin\,\theta_{2,1} \sin\,\theta_{3,1} = \cos(\theta_{2,1} + \theta_{3,1})$ and $\cos\,\theta_{2,1} \cos\,\theta_{3,1} + \sin\,\theta_{2,1} \sin\,\theta_{3,1} = \cos(\theta_{2,1} - \theta_{3,1})$, but cannot fall outside the interval determined by these endpoints. Depending on the values of the constraints, this interval might be much smaller than $[-1, 1]$, thereby reducing the uncertainty of the adversary.

The illustration on the right shows that some constraints yield a theoretical interval R which is strictly included in one of the $B = 3$ classification bins. For such constraints, the adversary can infer the target correlation bin with certainty. These constraints are of the two types. First, constraints that are close to 1 in absolute value result in high $\rho(X_1, X_2)$ belonging to the positive, green bin, when they have the same sign and to the negative, blue bin, when they have opposite signs. Second, when one of the constraints is close to 0 while the other is close to 1 in absolute value, $\rho(X_1, X_2)$ is necessarily in the low, orange bin.



To provide a precise, mathematical description of these regions, we analyse in detail the sets of constraints $(\rho(X_1, Y), \rho(X_2, Y))$ satisfying that there is only one possible bin for $\rho(X_1, X_2)$. Let $0 \leq \theta_1, \theta_2 \leq \pi$ be such that $\rho(X_1, Y) = \cos\theta_1$ and $\rho(X_2, Y) = \cos\theta_2$. As a reminder, the $B = 3$ classification bins are $[-1, -1/3)$, $[-1/3, 1/3)$, and $[1/3, 1]$ and the interval attainable by $\rho(X_1, X_2)$ is $[\cos(\theta_1 + \theta_2), \cos(\theta_1 - \theta_2)]$. Thus depending on the bin, θ_1 and θ_2 must satisfy the following inequalities:

- **Negative bin:** $\cos(\theta_1 - \theta_2) \leq -\frac{1}{3}$. It follows that $\theta_1 - \theta_2 \geq \arccos(-\frac{1}{3})$ if $\theta_1 \geq \theta_2$ and $\theta_2 - \theta_1 \geq \arccos(-\frac{1}{3})$ otherwise.

- **Low bin:** $-\frac{1}{3} \leq \cos(\theta_1 + \theta_2) \leq \cos(\theta_1 - \theta_2) < \frac{1}{3}$. We distinguish 4 subcases:

  (I) $\theta_1 \geq \theta_2$ and $0 \leq \theta_1 + \theta_2 \leq \pi$. It follows that $\theta_1 + \theta_2 \leq \arccos\left(-\frac{1}{3}\right)$ and $\theta_1 - \theta_2 >$

$\arccos\left(\frac{1}{3}\right)$.

(II) $\theta_1 < \theta_2$ and $0 \leq \theta_1 + \theta_2 \leq \pi$. It follows that $\theta_1 + \theta_2 \leq \arccos\left(-\frac{1}{3}\right)$ and $\theta_2 - \theta_1 > \arccos\left(\frac{1}{3}\right)$.

(III) $\theta_1 \geq \theta_2$ and $\pi \leq \theta_1 + \theta_2 \leq 2\pi$. It follows that $\theta_1 + \theta_2 \geq \pi + \arccos\left(\frac{1}{3}\right)$ and $\theta_1 - \theta_2 > \arccos\left(\frac{1}{3}\right)$.

(IV) $\theta_1 < \theta_2$ and $\pi \leq \theta_1 + \theta_2 \leq 2\pi$. It follows that $\theta_1 + \theta_2 \geq \pi + \arccos\left(\frac{1}{3}\right)$ and $\theta_2 - \theta_1 > \arccos\left(\frac{1}{3}\right)$.

- **Positive bin:** $\frac{1}{3} < \cos(\theta_1 + \theta_2)$ It follows that $\theta_1 + \theta_2 < \arccos(\frac{1}{3})$ if $\theta_1 + \theta_2 \leq \pi$ and to $\theta_1 + \theta_2 > \pi + \arccos(-\frac{1}{3})$ if $\pi \leq \theta_1 + \theta_2 \leq 2\pi$.

In practice, we will derive the endpoints of the interval empirically, by sampling a large number of correlation matrices satisfying the correlation constraints. We describe next our matrix generation procedures.

6.3.4 Sampling correlation matrices under constraints

To generate a wide variety of correlation matrices satisfying the correlation constraints given by the adversary knowledge, we adapt an algorithm developed by Numpacharoen and Atsawarungruangkit [241]. The goal of the original algorithm is to sample valid correlation matrices (under no constraints). Algorithm 7 describes the procedure, which we include here for clarity and reproducibility. The insight is that each correlation coefficients $c_{i,j}$ can be expressed as $c_{i,j} = m_{i,j} + \cos\theta_{i,j} l_{i,j}$, where $m_{i,j}$ and $l_{i,j}$ only depend on $\theta_{p,q}$ for $1 \leq p \leq i$ and $1 \leq q < j$. This insight is used to generate valid correlation matrices by sampling the correlations in order, from top to bottom ($i = 1, \ldots, n$) and from left to right ($j = 1, \ldots, n$), uniformly within the bounds $m_{i,j} \pm l_{i,j}$ derived from the values previously sampled. The formulas on $m_{i,j}$ and $l_{i,j}$ can be derived (supposing without loss of generality that $i \geq j$) by developing the computation of $c_{i,j} = (BB^T)_{i,j} = B_{i,1:j-1}(B_{j,1:j-1})^T + b_{i,j}b_{j,j}$ and setting $m_{i,j} = \sum_{p=1}^{j-1}(\cos\ \theta_{i,p} \cos\ \theta_{j,p} \prod_{q=1}^{p-1} \sin\ \theta_{i,q} \sin\ \theta_{j,q})$ and $l_{i,j} = \prod_{q=1}^{j-1} \sin\ \theta_{i,q} \sin\ \theta_{j,q}$.

Algorithm 7 SAMPLECORRMATRIX[241]

1: **Inputs:**
   n: Number of variables.
2: **Output:**
   $C \in \mathbb{R}^{n \times n}$: A valid correlation matrix.
3: **Initialise:**
   $C \leftarrow 0$; $B \leftarrow 0$
4: // Randomly initialise the first column.
5: **for** $i \in \{2, \ldots, n\}$ **do**
6: $c_{i,1} \leftarrow \mathcal{U}(-1, 1)$ // $\cos\theta_{i,1}$
7: $b_{i,1} \leftarrow c_{i,1}$
8: $b_{i,j} \leftarrow \sqrt{1 - c_{i,1}^2}$ for $j \in \{2, \ldots, i\}$ // $\sin\theta_{i,1}$
9: **end for**
10: **for** $i \in \{2, \cdots, n\}$ **do**
11: **for** $j \in \{1, \cdots, i\}$ **do**
12: // Compute the bounds $m_{i,j} \pm l_{i,j}$.
13: $m_{i,j} \leftarrow B_{i,1:j-1}(B_{j,1:j-1})^T$
14: $l_{i,j} \leftarrow b_{i,j} b_{j,j}$
15: // Sample uniformly within bounds.
16: $c_{i,j} \leftarrow \mathcal{U}(m_{i,j} - l_{i,j}, m_{i,j} + l_{i,j})$
17: // Update B.
18: $aux \leftarrow \frac{c_{i,j} - m_{i,j}}{l_{i,j}}$ // cos $\theta_{i,j}$.
19: $b_{i,j} \leftarrow b_{i,j} * aux$
20: $b_{i,k} \leftarrow \sqrt{1 - aux^2}$ for $k \in \{j+1, \ldots, n\}$
21: **end for**
22: **end for**
23: $C \leftarrow C + C^T + I_n$

To adapt this algorithm for generating matrices under constraints given by the adversary knowledge, we notice that the coefficients of the first column of B ($\cos\theta_{i,1}$ for $i = 2, \ldots, n-1$) are identical to those of C and are free parameters. We start by reordering the variables to $Y, X_1, \ldots, X_{n-1}$ to generate $\rho(X_1, X_2)$ first (and explain below why). Under **S2**, we set the coefficients equal to the correlation constraints $(\rho(X_i, Y))_{i=1,\ldots,n-1}$. This amounts to replacing line 6 from Algorithm 7 (highlighted in red) with $c_{i,1} \leftarrow \rho(X_{i-1}, Y)$. Under **S1**, we will set the first two coefficients equal to the correlation constraints $\rho(X_1, Y)$ and $\rho(X_2, Y)$, initializing the others uniformly at random within $[-1, 1]$. We replace line 6 from Algorithm 7 correspondingly.

We then shuffle the variables 3 to $n-1$ by applying a random permutation. Indeed, naïvely applying Algorithm 7 would generate the correlations using the default variable ordering (in our case, $Y, X_1, \ldots, X_{n-1}$). While the correlation generated first, $\rho(X_1, X_2)$, is uniformly dis-

Algorithm 8 SampleCorrMatrix-AttackS1

Inputs:
n: Number of variables.
$\rho(X_1, Y)$ and $\rho(X_2, Y)$: Attacker knowledge.
Output:
$C \in \mathbb{R}^{n \times n}$: A valid correlation matrix satisfying $c_{1,n} = \rho(X_1, Y)$ and $c_{2,n} = \rho(X_2, Y)$.
$C \leftarrow$ SampleCorrMatrix$(n, constraints)$
// Reorder variables as $X_1, X_2, X_{\sigma(3)}, \ldots, X_{\sigma(n-1)}, Y$.
$\sigma \leftarrow [2, 3] + random\_permutation([4, \ldots, n]) + [1]$
$C \leftarrow reorder\_columns(reorder\_rows(C, \sigma), \sigma)$

Algorithm 9 SampleCorrMatrix-AttackS2

Inputs:
n: Number of variables.
$(\rho(X_i, Y))_{i=1,\ldots,n-1}$: adversary knowledge.
Output:
$C \in \mathbb{R}^{n \times n}$: A valid correlation matrix satisfying $c_{i,n} = \rho(X_i, Y)$, for $i = 1, \ldots, n-1$.
// Randomly shuffle the constraints.
$\sigma \leftarrow [1, 2] + random\_permutation([3, \ldots, n-1])$
$constraints \leftarrow \sigma(constraints)$
$C \leftarrow$ SampleCorrMatrix$(n, constraints)$
// Reorder the variables as $X_1, \ldots, X_{n-1}, Y$.
$\sigma' \leftarrow [2, \ldots, n] + [1]$
$C \leftarrow reorder\_columns(reorder\_rows(C, \sigma'), \sigma')$
// Revert the shuffling of constraints.
$\sigma_n \leftarrow \sigma^{-1} + [n]$
$C \leftarrow reorder\_columns(reorder\_rows(C, \sigma_n), \sigma_n)$

tributed within its boundaries, the correlation generated last, $\rho(X_{n-2}, X_{n-1})$, is not, because the previously sampled correlations restrict its range. More specifically, its conditional distribution follows a bell-shaped curve. Shuffling the variables reduces this bias [241]. We choose to keep the target pair X_1 and X_2 in the first position, to guarantee a uniform prior over the distribution of the target correlation $\rho(X_1, X_2)$. Indeed, we assume the adversary does not know how $\rho(X_1, X_2)$ is distributed within its boundaries. Furthermore, we do not want to bias the values towards the centre of the interval, as would be the case if the value was not generated first. Algorithm 9 details how we carefully sequence the shuffling, the column reordering, and the inversion of the shuffling for scenario **S2**.

Finally, Algorithm 10 details our procedure to sample the unknown value $\rho(X_1, X_2)$ for scenario **S3**, when all the other correlations are known. We start by placing the target pair (X_1, X_2) in

Algorithm 10 SampleCorrMatrix-AttackS3

Inputs:
n: Number of variables.
C: adversary knowledge having $c_{1,2}$ and $c_{2,1}$ filled with a dummy value.
Output:
$D \in \mathbb{R}^{n\times n}$: A valid correlation matrix satisfying $d_{i,j} = c_{i,j}$, $\forall (i,j) \in \{1,\dots,n\}^2 \setminus \{(1,2),(2,1)\}$.
// Reorder the variables as $Y, X_{n-1}, \dots, X_1$.
$\sigma \leftarrow [n, \dots, 1]$
$C' \leftarrow reorder\_columns(reorder\_rows(C,\sigma),\sigma)$
$B'B'^T \leftarrow$ **CholeskiDecomposition**$(C'_{1:n-1,1:n-1})$ // $B' \in \mathbb{R}^{(n-1)\times(n-1)}$
$B \leftarrow 0$; $B_{1:n-1,1:n-1} \leftarrow B'$ // $B \in \mathbb{R}^{n\times n}$
$b_{n,1} \leftarrow c'_{n,1}$ // Start filling the last row of B.
for $i \in \{2, \cdots, n-2\}$ **do**
    $b_{n,i} \leftarrow b'_{n,i} - B_{i,1:i}(B_{n,1:i})^T / b_{i,i}$
end for
$b_{n,n-1} \leftarrow \sqrt{1 - \sum_{j=1}^{n-2} b_{n,j}^2}$
// Compute the boundaries $m \pm l$ of $c'_{n,n-1}$.
$m, l \leftarrow B_{n-1,1:n-1}(B_{n,1:n-1})^T, b_{n-1,n-1}b_{n,n-1}$
// Sample uniformly within the bounds.
$c'_{n,n-1} = c'_{n-1,n} \leftarrow \mathcal{U}(m-l, m+l)$
// Reorder the variables as $X_1, \dots, X_{n-1}, Y$.
$D \leftarrow reorder\_columns(reorder\_rows(C',\sigma))$

the last position. Denote by C' the partial correlation matrix after reordering the variables (filling the unknown $c'_{n,n-1}$ with a dummy value). We sample the last coefficient $c'_{n,n-1}$ uniformly within its bounds, which we determine by reverse-engineering part of the spherical parametrisation B. More specifically, we compute all the low triangular coefficients except for $b_{n,n-1}$ and $b_{n,n}$, which cannot be determined absent knowledge of $c'_{n,n-1}$. This allows us to compute the bounds of $c'_{n,n-1}$ and sample it uniformly within them.

Alternative approaches. The spherical parametrisation of correlation matrices provides a principled and effective approach for sampling valid correlation matrices. In contrast, the characterisation given by properties **P1-P4** of Sec. 6.3.2 does not easily translate into an effective algorithm for constructing correlation matrices. In preliminary experiments, we framed the correlation matrix generation problem as a constrained optimisation program. We initialized a $n \times n$ matrix by setting each element uniformly at random between -1 and 1, then projected it into the space of positive semi-definite matrices having all diagonal entries equal to 1 and

all elements between -1 and 1. We used the DCCP library [300]. Due to numerical instability, the resulting matrices often did not satisfy property **P4**, as they had at least one negative eigenvalue (40% of the time for $n = 4$ variables and decreasing steadily with n). As a result, several instances of the optimisation program are required to generate one valid correlation matrix, and their number increases with the number of variables n. In constrast, Alg. 8 succeeds 100% of the time, regardless of the number of variables. Furthermore, our approach is 10 times faster than one instance of the optimisation program. We also found that the correlations generated by the optimisation program are less likely to cover the entire range of values that are theoretically attainable.

6.3.5 Sampling data given a correlation matrix

To generate shadow datasets whose correlations are approximately equal to the correlation constraints $V = \{(c_T)_{i,j}, (i,j) \in P\}$, where C_T denotes the correlation matrix of private dataset D_T, we use the Gaussian copula generative model. This is a probabilistic model parameterised by (1) the marginals of the variables $(F_i)_{1 \leq i \leq n}$ and (2) a positive semi-definite covariance matrix Σ yielding the following joint distribution: $\Pr(X_1 \leq x_1, \ldots, X_n \leq x_n) = \Phi_\Sigma\big(\Phi^{-1}(F_1(x_1)), \ldots, \Phi^{-1}(F_n(x_n))\big)$, where Φ_Σ is the cumulative distribution function (CDF) of the Gaussian multivariate distribution $\mathcal{N}(0, \Sigma)$:

$$\Phi_\Sigma(x_1, \ldots, x_n) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_n} \frac{e^{-\frac{1}{2} x^T \Sigma^{-1} x}}{\sqrt{(2\pi)^n \det(\Sigma)}} \tag{6.7}$$

and Φ is the CDF of a one-dimensional standard normal $\mathcal{N}(0, 1)$. Algorithm 11 describes the procedure to sample from the Gaussian copulas.

Consistent with the rest of our methodology, we will use correlation, rather than covariance matrices to parametrise the Gaussian copulas. Correlation matrices are a subset of covariance matrices that have only values of 1 on the diagonal.

To generate synthetic datasets D matching the correlation constraints and the one-way marginals of the private dataset D_T, we distinguish between two cases:

Algorithm 11 SAMPLEFROMGAUSSIANCOPULAS

Inputs:
n: Number of variables.
$\Sigma \in \mathbb{S}_+^d$: A valid covariance matrix.
$F_1, \ldots, F_n$: One-way marginals.
Output:
A sample $z = (z_1, \ldots, z_n)$ from a Gaussian copula distribution parametrised by $F_1, \ldots, F_n$ and Σ.
$A \leftarrow$ **CholeskyDecomposition**(C) // $C = A^t \cdot A$
$Z \sim \mathcal{N}(0, I_n)$
$X \leftarrow A^T Z$
for $i = 1, \ldots, n$ **do**
    $z_i \leftarrow F_i^{-1}(X_i)$
end for
$z \leftarrow (z_1, \ldots, z_n)$

1. When the variables are all standard normals, i.e., $F_i = \Phi, i = 1, \ldots, n$, the correlations of the distribution output by the Gaussian copula are identical to the parameter Σ. We first generate a correlation matrix C using the algorithm corresponding to the attack scenario (e.g., Alg. 9 for **S2**), then set $\Sigma = C$ to sample a dataset whose empirical correlation approximately match C.

2. When the variables have arbitrary one-way marginals, the correlations of the distribution output by the Gaussian copula parameterised by these marginals are not necessarily equal to the parameter Σ [349]. This means that the adversary must not directly parameterise the Gaussian copula by setting $\Sigma = C$, but instead *must modify the correlation constraints* $V = \{(c_T)_{i,j}, (i,j) \in P\}$ (with C_T denoting the correlation matrix of private dataset D_T) to a new set of constraints V', generate a correlation matrix C' under the constraints given by V' (instead of V), and only then set $\Sigma = C'$. V' is chosen such that the empirical correlations computed on the synthetic datasets are approximately equal to V and thus match the correlation constraints.

   Algorithm 12 details a simple, yet very effective heuristic to modify the correlation constraints. More specifically, it determines V' such that, on average, shadow datasets D generated from matrices matching V' minimise the difference $\max_{(i,j)\in P} |E[C(D)_{i,j}] - (c_T)_{i,j}|$ between their empirical correlations $C(D)$ and the correlation constraints. For concise-

ness, we focus on the default scenario **S2**, where $V = (\rho(X_i, Y))_{i=1,\dots,n-1}$ (of which scenario **S1** is a particular case).

Algorithm 12 ModifyCorrelationConstraints

Inputs:
n: Number of variables.
$V = (\rho(X_i, Y))_{i=1,\dots,n-1} \in \mathbb{R}^{n-1}$: Correlation constraints.
$(F_i)_{i=1,\dots,n}$: One-way marginals.
S: Number of shadow datasets used to estimate the gap.
N_D: Size of the shadow datasets.
e: Error tolerance.
M: Maximum number of iterations.
Output:
$V' \in \mathbb{R}^{n-1}$: Modified correlation constraints.
Initialise:
$V' \leftarrow V$
for $m = 1, \dots, M$ **do**
  $\bar{V} \leftarrow zeros(n-1)$
  for $s = 1, \dots, S$ **do**
    $C \leftarrow$ SampleCorrMatrix-AttackS2(n, V')
    // Shadow dataset whose correlations approximately satisfy V'.
    $D^s_{\text{shadow}} \leftarrow [\text{SampleFromGaussianCopulas}(n, C, F_1, \dots, F_n) \text{ for } d = 1, \dots, N_D]$
    $C^s \leftarrow correlation\_matrix(D^s_{\text{shadow}})$
    $\bar{V}_i \leftarrow \bar{V}_i + \frac{C^s_{i,n}}{S}$ for $i = 1, \dots, n-1$
  end for
  $gap \leftarrow \max_{i=1,\dots,n-1} |\bar{V}_i - V_i|$
  if $gap < e$ **then**
    break
  else
    // Update the constraints.
    $V'_i \leftarrow max(-1, min(1, V'_i + \frac{V_i - \bar{V}_i}{2})$
  end if
end for

6.3.6 Model-based attack

Our model-based attack exploits access to a machine learning model $\mathcal{M}_T$ trained on the target dataset D_T in order to extract the target correlation. We extend the shadow modelling technique described in Sec. 5.3 to infer correlations and to use synthetic shadow datasets rather than real records from a similar distribution. In particular, our synthetic datasets are generated

so as to satisfy the correlations and one-way marginal constraints imposed by the adversary knowledge.

First, we generate K shadow correlation matrices $C^1, \ldots, C^K$ satisfying V when the marginals are standard normals (and V' for arbitrary marginals). In each scenario, the correlation matrices are sampled using the corresponding algorithm from Sec. 6.3.4. For each shadow correlation matrix C^k, we generate a synthetic shadow dataset D^k_{shadow} whose correlations approximately match V and whose marginals match $F_i, i = 1, \ldots, n$ as described in Sec. 6.3.5.

Second, we train a shadow model $\mathcal{M}^k$ on shadow dataset D^k_{shadow}, for every $k = 1, \ldots, K$. The models are trained using the same algorithm and hyperparameters as the target model, but a different seed. We present in Sec. 6.8 results of our attack when the adversary knows the seed.

Third, we generate a synthetic *query dataset* D_{query} in the same way as above, and use it to extract black-box features from the shadow models. More specifically, after the K shadow models have been trained, we pass the records of D_{query} through the shadow model $\mathcal{M}^k$ for $k = 1, \ldots, K$, and retrieve the confidence scores for the first $L-1$ classes (the last one is redundant with the others as they all sum to 1). Denoting by Q the size of D_{query}, we thus extract a total of $Q \times (L-1)$ features from each model. In this work, we assume for simplicity that the models are binary classifiers ($L = 2$), meaning that we extract Q features from each model. We concatenate them to form the *output feature vectors* $O_k, l = 1, \ldots, K$. Our hypothesis is that these features are likely to encode information about the dataset correlations, e.g. that a model makes more confident predictions on records resembling its own training dataset.

Fourth, given a dataset D, we denote by $C(D)$ the correlation matrix computed on the dataset. We label each feature vector O_k with the correlation $L_k = C(D^k_{shadow})_{1,2}$ of the corresponding shadow dataset, discretised over B classes. We then train a meta-classifier $\mathcal{A} : \mathbb{R}^Q \to \{1, \ldots, B\}$ on $D_{\text{meta}} = \{(O_k, L_k), k = 1, \ldots, K\}$ to infer the correlation given features extracted from the model.

Finally, we apply the meta-classifier to the target model $\mathcal{M}_T$. The adversary's guess is the output class $\mathcal{A}(O_T)$ predicted by the meta-classifier on the feature vector O_T, consisting of the

outputs of $\mathcal{M}_T$ on the query dataset D_{query}.

6.4 Experimental setup

6.4.1 Target models

The target models we study in this work are the Logistic Regression (LR) and the Multilayer Perceptron (MLP). We use the LR implementation provided by the Scikit-learn 0.24.1 library with default parameters. We implement the MLP models in Pytorch 1.10.0 using an architecture that consists of two hidden layers of sizes 20 and 10 with ReLU nonlinearity. In the experiments on synthetic data (Sec. 6.5.1- 6.5.3), we train the models using gradient descent. In the experiments on real-world datasets (Sec. 6.5.4), we instead train the models using mini-batch gradient descent with a batch size of 128, as this leads to better target model accuracy. The models are trained on 90% of the samples for up to 100 epochs, stopping the models after 5 epochs of non-improving accuracy on the remaining 10%. We use a learning rate of $\eta = 0.05$.

6.4.2 Datasets

We use three datasets in our real-world evaluation, presented in Sec. 6.5.4.

Fifa19 [128] contains 18207 records of 89 attributes each, describing physical attributes and performance statistics of football players. We discard the categorical attributes, keeping the continuous and discrete ordinal variables. We select the *Value* attribute, representing the amount of money a club payed for a player, as the output variable, discarding the other attributes predictive of it that would make classification trivial (e.g., *Wage*). We then remove the records with missing values, leaving us with 15917 records. We binarise the output variable by mapping it to $Y = 1$ if larger than the median value (> 0.74M) and to $Y = 0$ otherwise. Finally, we remove the duplicate attributes, which leaves us with 53 input attributes.

Communities and Crime [268] contains 2215 records of 147 attributes each, describing

socio-economic, law enforcement, and crime statistics on communities in the US. We select the number of murders as the output variable of our machine learning models, discarding the other columns predictive of it (i.e., relating explicitly to crime). As input attributes, we select the 101 continuous and discrete ordinal attributes from the dataset that do not have missing values, discarding the rest. We binarise the number of murders by mapping it to $Y = 1$ if is at least 1, and to $Y = 0$ otherwise.

Musk (v2) [100] contains 6598 records of 166 attributes each, describing different conformations of 165 molecules, together with a human expert-assigned label of "musk" or "non-musk". We train the models to classify conformations as "musk" ($Y = 1$) or "non-musk" ($Y = 0$). The dataset being heavily imbalanced with respect to the two classes, we balance it for simplicity, leaving us with 2034 records.

6.4.3 Attack parameters

Unless otherwise specified, throughout the experiments we will consider the default attacker **S2**. Given a target model $\mathcal{M}_T$, we train a meta-classifier on outputs extracted from $k = |D_{\text{meta}}|$ shadow models. We generate $K = 5000$ shadow datasets for target models $\mathcal{M}_T$ trained on $n = 3, 4$ and 5 variables and $K = 10000$ shadow datasets for $n = 6$ to 10 variables. In the experiments on synthetic data, the target and shadow models are trained on 1000 samples and the target model test accuracy is reported on 500 unseen samples. In the experiments on real-world datasets, the target models are trained on all the samples available ($|D_T| = 2215$ for Communities and crime and $|D_T| = 2034$ for Musk), except for Fifa19, where we sample $|D_T| = 2000$ records uniformly without replacement to keep the dataset sizes similar. The shadow datasets are always generated to have a number of samples equal to the target dataset size $|D_T|$. We run the ModifyCorrelationConstraints heuristic (Algorithm 12) using parameter values $S = 100$ and $N_D = |D_T|$, for a maximum of $M = 10$ iterations and with an error tolerance of $e = 0.01$. We set the size of the query dataset D_{query} equal to $Q = 100$, and discuss in Sec 6.5.3 the impact of this parameter.

We use a Logistic Regression as the meta-classifier for LR models, as it leads to similar accuracy

as the MLP but takes less time to train. We use an MLP as the meta-clasifier for MLP models, trained the same as stated in Sec. 6.4.1 except that we use a batch size of 128, a learning rate of $\eta = 0.001$, $\mathbb{L}_2$ weight decay of 0.01, and early stopping after 10 epochs of non-improving accuracy on 10% of held-out samples.

6.5 Results

In this section, we first quantify how much information is leaked by different types of models, comparing the performances of our model-based and model-less attacks. Second, we analyse, under three different attack scenarios, the impact of the number of variables in the dataset on the performance of the attack. Third, we consider several possible mitigation strategies. Finally, we validate the effectiveness of our attack against real-world datasets.

6.5.1 Impact of constraints for $n = 3$ variables

To characterise how the correlation constraints impact the performance of the attack, we start with a simple setup having only two input variables and an output variable ($n = 3$).

We consider all possible pairs of constraints $\rho(X_1, Y)$ and $\rho(X_2, Y)$ by dividing the grid $[-1, 1] \times [-1, 1]$ in 200×200 equal cells. In each cell, we evaluate the accuracy of inferring the target correlation $\rho(X_1, X_2)$ over T' pairs of constraints drawn uniformly in that cell. Each pair is used to generate, in order: (1) a target correlation matrix C_T, (2) a target dataset D_T of 1000 samples using Gaussian copulas parameterised by C_T and standard normal marginals, and (3) a target model $\mathcal{M}_T$ trained on D_T to perform the binary classification task $Y > 0$. Given access to $\mathcal{M}_T$ and to the pair of constraints, the adversary goal is to infer $(c_T)_{1,2} = \rho(X_1, X_2)$. Executing our attack independently on T' targets would require us to generate K shadow models for each target of each cell, which is computationally infeasible. Instead, we train one meta-classifier per cell, and evaluate it using 5-fold cross-validation and setting $T' = 1500$.

Fig. 6.1A shows an adversary with no access to the model can infer the target correlation

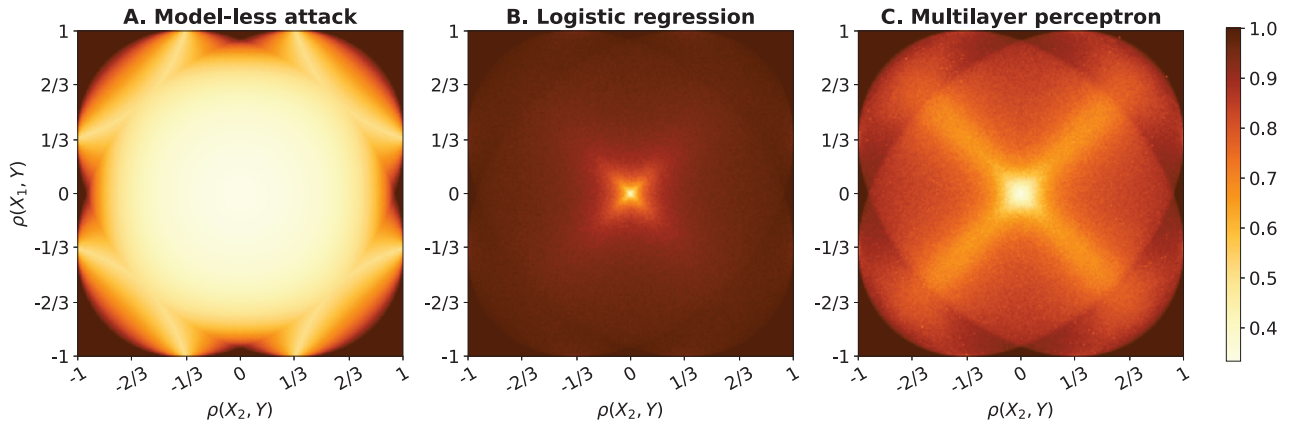


Figure 6.1: **Accuracy of the correlation inference attack on $n = 3$ variables and different target models.** We consider all the possible values of the correlation constraints $\rho(X_1, Y)$ and $\rho(X_2, Y)$ using a fine discretisation of the grid (200×200 cells). The colour of each cell corresponds to the accuracy of our correlation inference attack on synthetic datasets. The datasets are sampled using Gaussian copulas parametrised by correlation matrices that satisfy the constraints $\rho(X_1, Y)$ and $\rho(X_2, Y)$ defining the cell.

$\rho(X_1, X_2)$ 56.0% of the time on average over all the cells, significantly better than random (33.3%). These results also confirm the theoretical analysis presented in Sec. 6.3.3 with our model-less attack achieving perfect accuracy in the corners of the grid and the top centre region (with its symmetric counterparts). The accuracy of the model-less attack slowly then decreases as the constraints move to regions of higher uncertainty, in which more than one bin is possible.

Fig. 6.1B shows that black-box access to a Logistic Regression (LR) allows an adversary to greatly improve the accuracy of the attack, reaching 95.6% on average over all the cells.

As expected, the model-based attack achieves perfect accuracy in the regions where the model-less attack also does. Outside these regions, the model-based attack vastly outperforms the model-less attack. Its accuracy exceeds 90% everywhere, with the exception of the cross-like region centred in the origin where the models leak less information than elsewhere. At the centre of the cross ($|\rho(X_i, Y)| \leq 0.02, i = 1, 2$), where the input variables X_1 and X_2 are uncorrelated with the output variable Y, the accuracy of our attack is 47.2%. When there is little to no correlation between the inputs and the output variable Y ($\rho(X_i, Y) \simeq 0$), the target models do not learn much about the dataset. As for the arms of the cross, where $\rho(X_1, Y) \simeq \rho(X_2, Y) \ll 1$, the lower attack performance is due to the theoretical interval attainable by the target correlation $\rho(X_1, X_2)$ being close to $[-1, 1]$ and hence the adversary

having maximum uncertainty. Even in this scenario, the leakage from the model is much higher than the baseline.

Fig. 6.1C shows that multilayer perceptron (MLP) models are also vulnerable to our attack, achieving an accuracy of 82.3% on average over the cells. This shows that access to an MLP model greatly improves over the model-less attack, but slightly less than the LR. This may seem surprising at first, since our MLP, with its 292 learnable weights, has a much higher capacity than the LR with its 3 weights and thus a higher ability to retain information about the dataset. We believe that this gap is due to the fact that the non-convex loss used to train MLP models has a (potentially large) number of local minima [69]. This means that an adversary's shadow models are unlikely to reach the same local minima as the target model, increasing the adversary's uncertainty. We refer the reader to Sec. 4.8 for results of our attack when we remove the uncertainty caused by the difference in randomness. We also find that the cross-like behaviour is exacerbated on the MLP: the centre of the cross is larger, reaching an average accuracy of 48.1% for $|\rho(X_1, Y)| \leq 0.1$.

White-box results. We further evaluate the performance of white-box correlation inference attacks using the model weights as features. The weights of the Logistic Regression are its coefficients (including the bias). For the MLP, we explore two options: (1) using the *raw weights* (coefficients of the linear layers, including the bias), flattened and concatenated into a single vector and (2) the *canonical weights*, extracted in the same way from the model after sorting the neurons in each layer according to the sum of weights [131].

Fig. 6.2 shows that the weights of Logistic Regression models yield similar performance to the confidence scores (95.1% vs 95.6%). This is likely due to the small number of weights (three in total), whose information we believe to be redundant with the one contained in the confidence scores extracted from $Q = 100$ records.

Fig. 6.3 shows that the raw weights of MLP models do not leak more information than the model-less attack (55.0%). This negative result is in line with previous findings from the property inference literature [131, 365] and can be attributed to the permutation equivalence property of MLPs [131], whereby neurons of internal layers can be permuted without changing

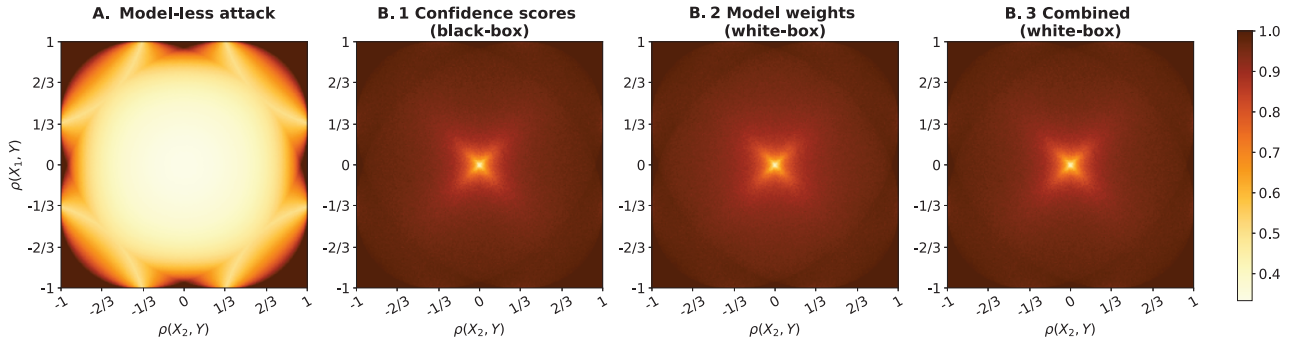


Figure 6.2: **Logistic Regression: Comparison between black-box and white-box correlation inference attacks.** We report results for white-box attacks using as features the model weights (B. 2) or the combined weights and confidence scores (B. 3).

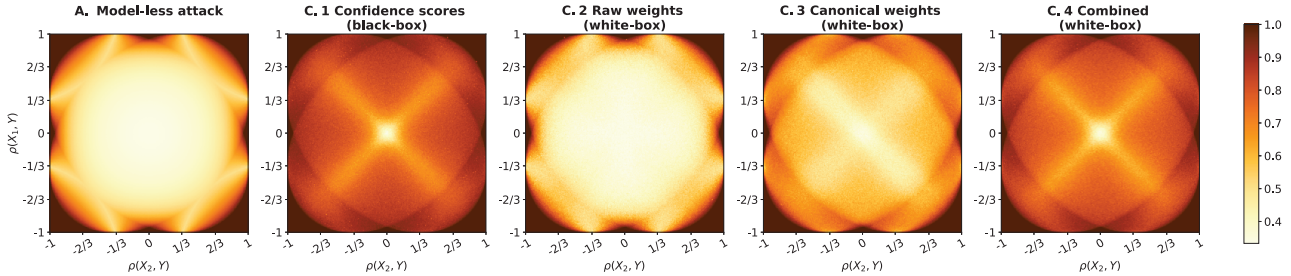


Figure 6.3: **Multilayer perceptron (MLP): Comparison between black-box and white-box correlation inference attacks.** We report results for white-box attacks using as features the raw model weights (C. 2), the *canonical* weights (mapped to a canonical representation [131], C. 3), and the combined canonical weights and confidence scores (C. 4).

the underlying function. As a result, a meta-classifier operating on inputs in their default ordering (with $k!$ equivalent orderings being possible in each layer, where k is the number of neurons in the layer) needs to learn a more difficult task. We show that the canonical weights yield much better performance (65.5%), albeit lower than the confidence scores.

On both models, combining the model weights and the confidence scores (Fig.6.2-B.3 and Fig. 6.3-C.4) does not yield better results than using the confidence scores alone.

We refer the reader to Sec. 6.8 for results of our white-box correlation inference attacks on MLP models when the attacker is assumed to know the seed used to initialise the target model's training algorithm.

6.5.2 Increasing the number of variables n

Next, we compare the performances of our attack against Logistic Regression and MLP models when increasing the number of variables n, under three different attack scenarios.

For each $n = 3, \ldots, 10$, we sample 1000 target correlation matrices $C_T \in \mathcal{C}$ using Algorithm 7. To obtain a balanced distribution over the target correlations, we generate the value $\rho(X_1, X_2)$ first, as explained in Sec. 6.3.4. Then for each C_T, we sample a synthetic target dataset D_T (parametrised by C_T and standard normal marginals) and train a target model $\mathcal{M}_T$ on it.

Fig. 6.4**S2** shows that the adversary is consistently and significantly better than the model-less attack. The performance of our attack decreases slowly with the number of variables in the dataset, from 96.1% ($n = 3$) to 70.4% ($n = 6$) and 58.2% ($n = 10$) when applied to LR models, while the model-less attack has a roughly constant accuracy of 49.3% for all values of n. This decrease when n increases is expected. Indeed, first, the number of unknown correlations increases quadratically with n, which increases the uncertainty of the adversary. Second, the more variables a model is trained on, the less likely our target variables X_1 and X_2 are to be useful for the prediction task at hand. This in turn means the model might learn encode less information about $\rho(X_1, X_2)$.

Fig. 6.5 shows that when at least one of the constraints $\rho(X_1, Y)$ or $\rho(X_2, Y)$ is large in absolute value (meaning that X_1, X_2, or both are good predictors of the output variable), our attack performs better. For instance, on $n = 6$ variables our attack reaches 90.6% when $\max(|\rho(X_1, Y)|, |\rho(X_2, Y)|) \geq 0.8$, but only 62.7% when $\max(|\rho(X_1, Y)|, |\rho(X_2, Y)|) \leq 0.2$.

Interestingly, the difference in vulnerability between Logistic Regression and MLP models reduces as the number of variables in the dataset n increases, with our attack achieving similar accuracy against the two for $n \geq 6$. We rule out the possibility of models learning the same decision boundary, as the MLP models are, on average, 1 to 4% less accurate, with the gap in accuracy increasing with n. We believe this could instead be due to MLP models's training algorithm becoming more stable as n increases, i.e., the loss having fewer local minima, as more input variables are likely to be predictive of the output Y. We obtain similar trends on $B = 5$

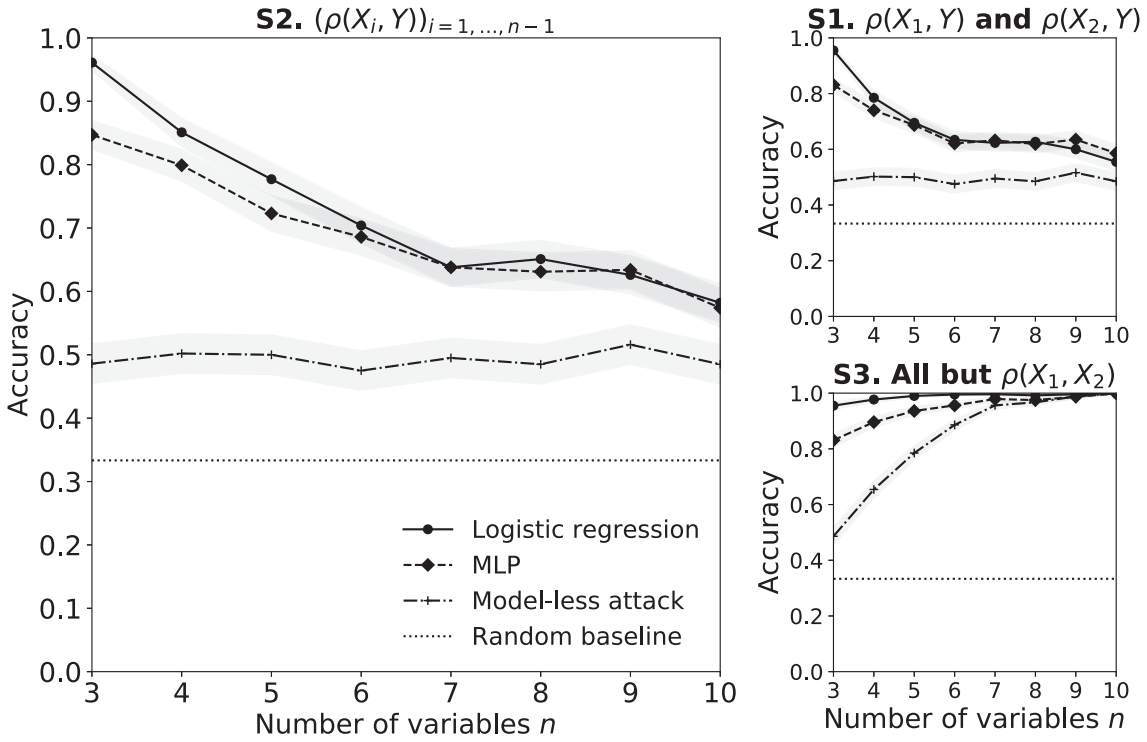
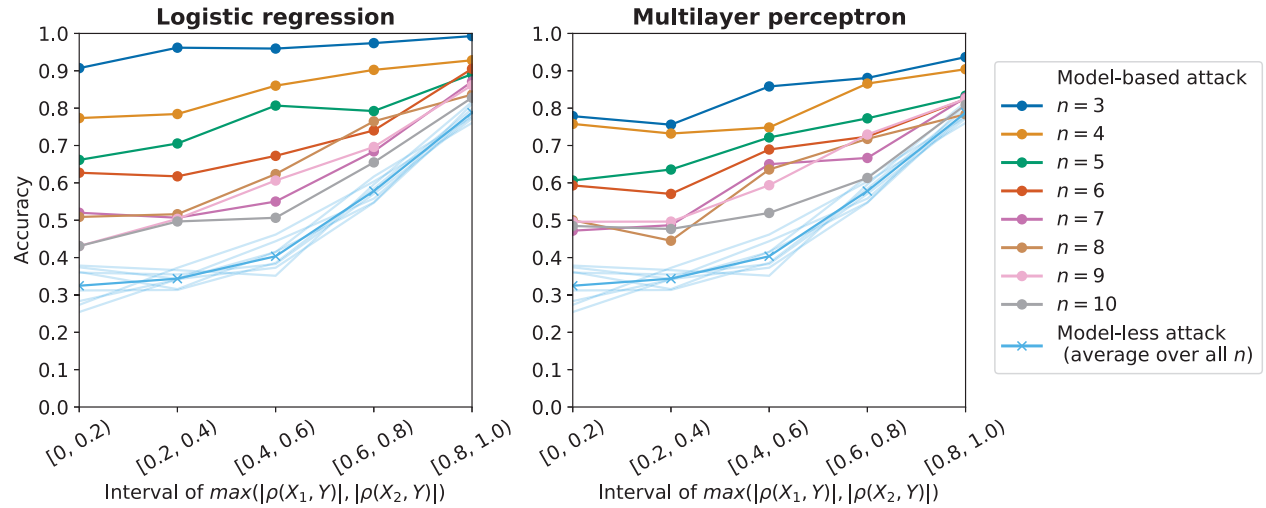


Figure 6.4: **Attack accuracy for different scenarios and number of variables in the dataset** n. The accuracy (with 95% confidence interval) is computed over 1000 target models.

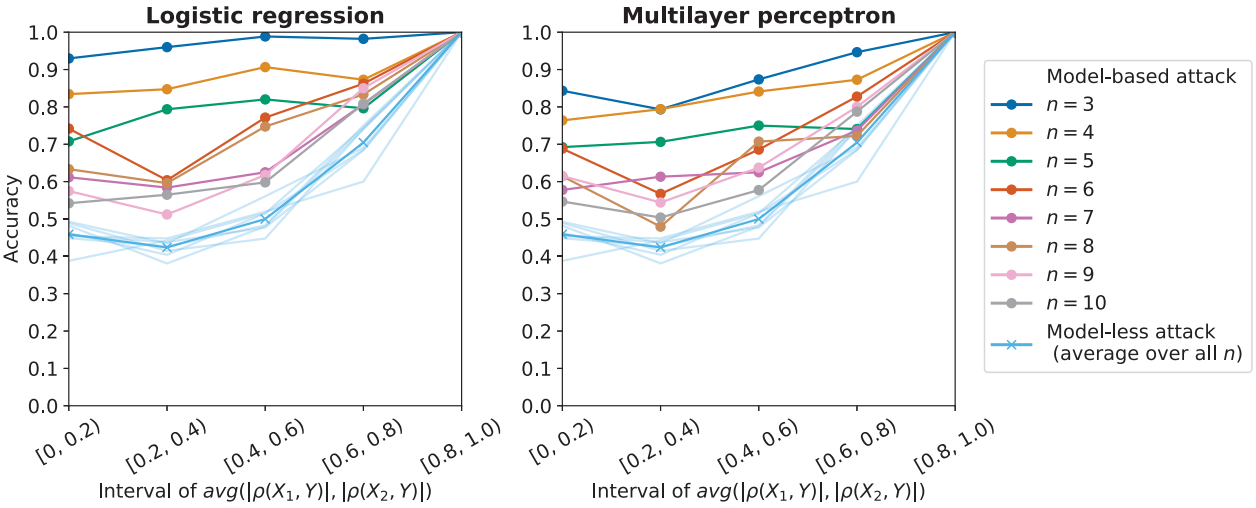
classification bins, as shown on Fig. 6.6.

Fig. 6.4**S1** shows that reducing the knowledge of correlations to only $\rho(X_1, Y)$ and $\rho(X_2, Y)$, only slightly decreases the attack accuracy. More specifically, the accuracy is 78.5% when $n = 3$ vs 85.1% for **S2**, 63.4% when $n = 6$ vs. 70.4% for **S2**, and 55.5% when $n = 10$ vs. 58.2% for **S2**. This shows that correlation inference attacks are a concern even when facing a weak adversary.

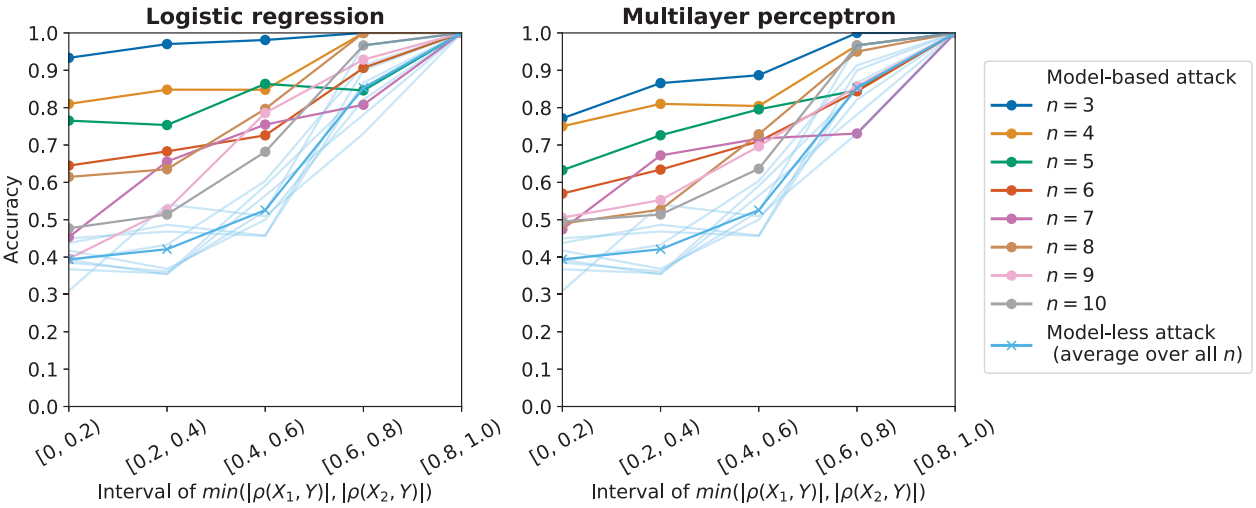
We complete our understanding of the impact of uncertainty on the attack by additionally considering a very strong adversary, who has knowledge of all the correlations but the target $\rho(X_1, X_2)$. Fig. 6.4**S3** shows that the accuracy of our attack now increases as n increases. We believe this is because the fraction of unknown correlations now reduces quadratically with n. More specifically, our attack improves as n increases, from 95.5% ($n = 3$) to 99.5% ($n = 6$) and 99.9% ($n = 10$) on Logistic Regression models. We obtain similar results on MLP models, as our attack improves from 83.1% ($n = 3$) to 95.6% ($n = 6$) and 99.9% ($n = 10$). Interestingly, the strongest adversary can again exploit a higher vulnerability in LR models compared to MLP models. The model-less attack also increases with n, but achieves a lower accuracy than



(a) Impact of the largest constraint $\max(|\rho(X_1, Y)|, |\rho(X_2, Y)|)$



(b) Impact of the average constraint $(|\rho(X_1, Y)| + |\rho(X_2, Y)|)/2$



(c) Impact of the smallest constraint $\min(|\rho(X_1, Y)|, |\rho(X_2, Y)|)$

Figure 6.5: **Impact of the constraints on the attack accuracy for different number of variables n.** We compare the results of our model-based and model-less attacks on the Logistic Regression and MLP models.

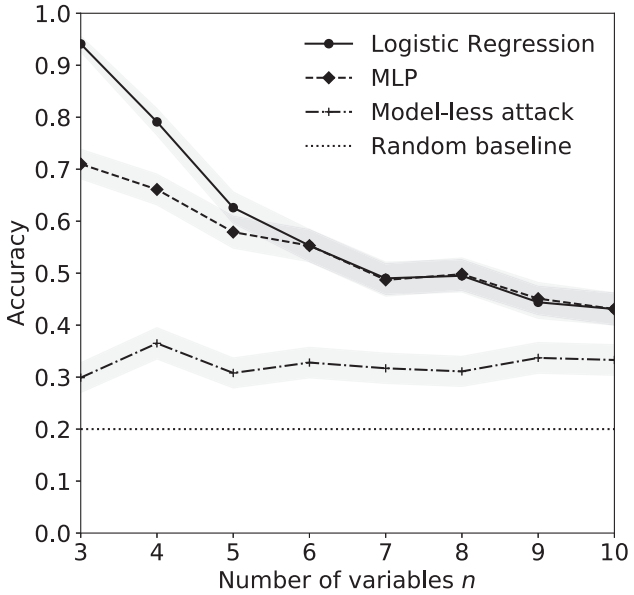


Figure 6.6: **Attack accuracy under scenario S2 and $B = 5$ correlation bins for different number of variables in the dataset** n. The accuracy (with 95% confidence interval) is computed over 1000 target models.

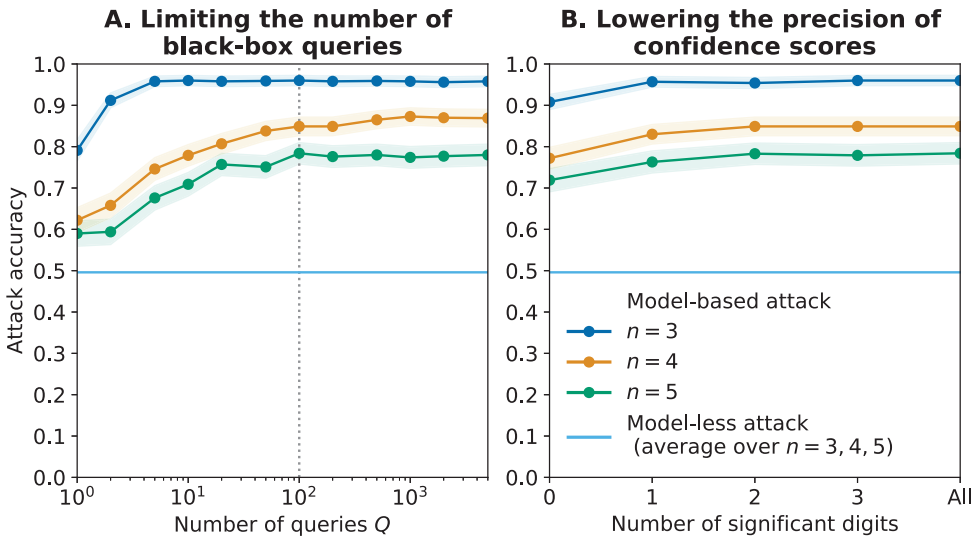
our model-based attack for $n \leq 8$.

6.5.3 Impact of mitigations

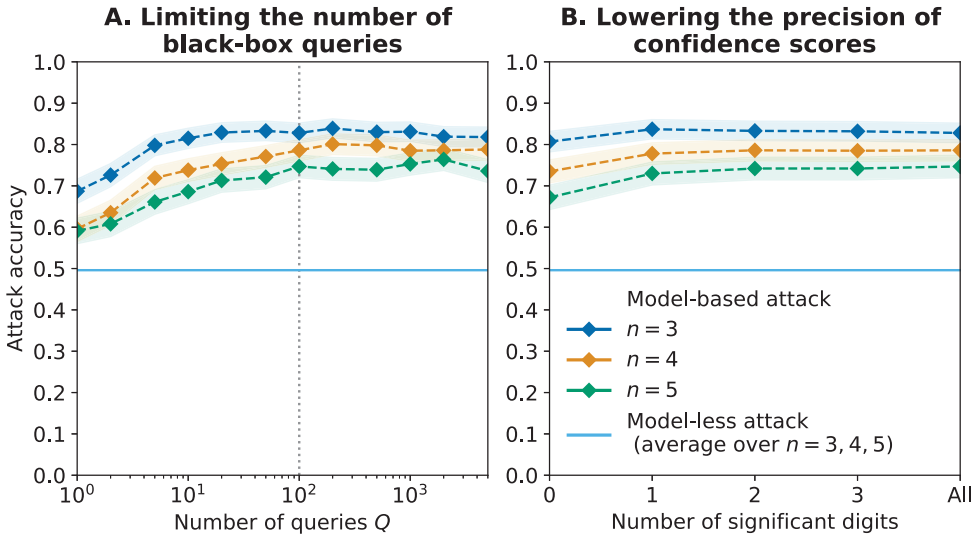
We analyse the effectiveness of three possible mitigations in preventing our correlation inference attack.

Limiting the number of queries could be put in place, e.g. in the machine learning as a service case, by the model developer in order to prevent the user from extracting too much information. We vary the number of queries Q made to an LR from 1 to 5000 and report the attack accuracy against the same targets as before. Fig. 6.7a (left) shows that this measure does not prevent our attack. Indeed only 5 queries are necessary to reach close to optimal accuracy for $n = 3$, 50 queries for $n = 4$ and 100 for $n = 5$. Even a single query is sufficient for the attack to reach 79.1% on $n = 3$ variables, 62.2% on $n = 4$, and 59.0% on $n = 5$, significantly higher than the model-less attack which only reaches 49.6%.

Lowering the precision of confidence scores is another popular mitigation deployed when releasing models, mainly by allowing the model to only output the class label [68]. Fig. 6.7a



(a) Logistic Regression



(b) Multilayer Perceptron

Figure 6.7: **Impact of mitigations on the accuracy of our attack.** We show the attack accuracy (with 95% confidence interval) over 1000 targets models for different number of variables n.

(right) shows that this measure has a very small impact on our attack, only decreasing its accuracy from 96.0% when all digits are made available to 90.8% when only the label is, on $n = 3$ variables (similarly, from 84.9% to 77.2% when $n = 4$ and from 78.4% to 71.9% when $n = 5$).

Taken together, our results suggests these mitigations to be ineffective in preventing our attack, with similar results on MLP models, as shown on Fig. 6.7b.

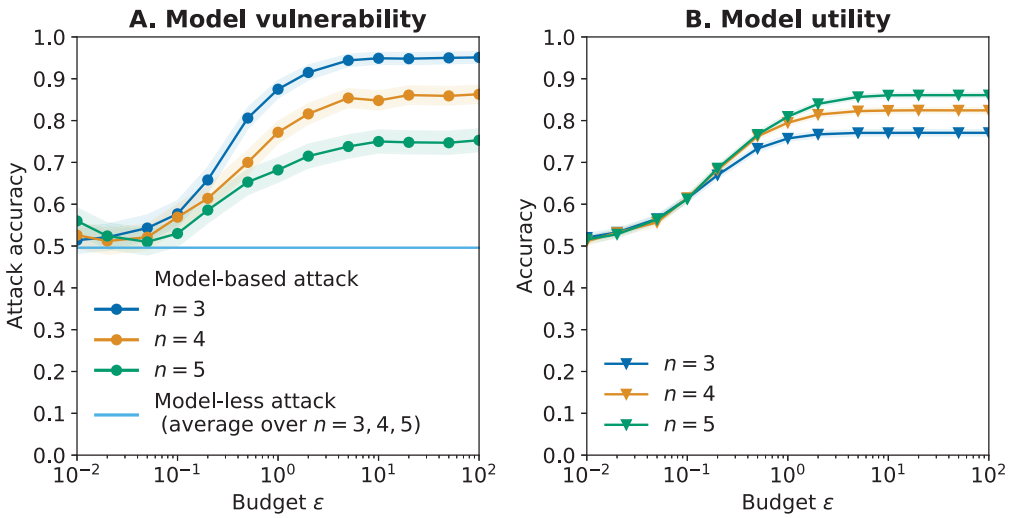


Figure 6.8: **Trade-off between the vulnerability of DP Logistic Regression models to our attack (left) and their utility (right).** We compute the metrics on 1000 models and report the 95% confidence intervals.

Differential privacy (DP) is a popular privacy definition requiring that the model should not depend too strongly on any one record, as controlled by ϵ [107]. DP protects individual privacy and ML models trained with DP guarantees have been shown to prevent individual-level attacks, including membership inference attacks. Although DP is not meant to protect population-level information such as correlations, the noise added might perturb the model enough to mitigate our attack. We train 1000 DP Logistic Regression models using the Diffprivlibrary[1] implementation of objective perturbation [62], and different values of ϵ.

Fig. 6.8 shows that the vulnerability of a model to our attack is strongly linked to its utility, i.e., accuracy. Indeed, for large value of ϵ, models are both useful and vulnerable. As we decrease ϵ, more noise is added which reduces both the vulnerability of models to our attack but also their utility. For small value of ϵ, our attack does not perform better than the model-less one, but at the cost of models losing any utility.

6.5.4 Real-world dataset evaluation

We evaluate the performance of our attack on the datasets described in Sec. 6.4.2. For each dataset, we select 100 data collections consisting of $n = 4$ columns, by sampling three different

[1] `https://diffprivlib.readthedocs.io/en/latest/index.html`

variables X_1, X_2, X_3 uniformly at random without replacement among the set of all triplets then adding the output variable Y. We report the mean and standard deviation of the attack accuracy averaged over 10 runs of 100 data collections each.

Table 6.1: **Results of our correlation inference attacks on three real-world datasets.** We abbreviate "Communities and Crime" to "C & C" for spacing reasons.

Number of bins	**Dataset**	**Random guess**	**Model-less attack**	**Model-based attack**	
				Logistic Regression	MLP
$B = 3$	Fifa19	33.3	60.2 (5.0)	91.2 (3.6)	78.8 (4.5)
	C & C	33.3	73.6 (2.7)	86.0 (3.6)	75.6 (4.0)
	Musk	33.3	67.8 (5.6)	82.0 (3.2)	56.3 (6.2)
$B = 5$	Fifa19	20.0	29.4 (5.1)	79.1 (3.6)	61.2 (6.1)
	C & C	20.0	27.6 (3.2)	70.6 (3.8)	56.0 (5.3)
	Musk	20.0	28.7 (4.5)	72.0 (5.5)	41.7 (6.4)

Contrary to the synthetic results, the correlations in our real-world datasets are not uniform. Instead they are, to a different extent depending on the dataset, skewed towards zero. This means that a large fraction of the target correlations belong to the "low" bin $[-1/3, 1/3)$. Beyond this it also means that the correlation constraints $\rho(X_1, Y) = \cos\theta_1$ and $\rho(X_2, Y) = \cos\theta_2$ are very small, thus leading to intervals $R = [\cos(\theta_1 + \theta_2), \cos(\theta_1 - \theta_2)]$ that are close to, but not equal to $[-1, 1]$ (e.g., $R = [-0.99, 0.99]$). These intervals thus encompass the "low" bin fully, and the other bins almost fully. For these two reasons, the model-less attack achieves on real-world datasets an "artificially" high accuracy. One solution would be to balance the test distribution. However, we instead prefer to show results for $B = 5$ bins. This indeed reduces the artificial advantage of the model-less attack while maintaining the generality of the evaluation.

Results. Table 6.1 shows that, despite its advantage, the model-less approach is outperformed by our model-based approach in five out of six cases. In Fifa19, it achieves an accuracy of 91.2% against LR models and 77.9% against MLP models, strongly outperforming the model-less attack which reaches 60.2%. Our attack similarly outperforms the model-less attack in the Communities and Crime dataset, achieving and accuracy of 86.0% against the LR models and 75.6% against the MLP models compared to the 73.6% reached by the model-less attack. Finally, while our attack achieves an accuracy of 82.0% on LR models, it is outperformed by the model-less attack when it comes to MLP models in the Musk dataset (56.3% vs 67.8%).

However, as soon as we increase the number of bins, removing the artificial advantage of the model-less attack, our attack vastly and consistently outperforms the model-less attack. On Fifa19, it achieves an accuracy of 79.1% against LR models and 61.2% against MLP models, strongly outperforming the model-less attack which reaches 29.4%. It similarly outperforms the model-less attack on the Communities and Crime and Musk datasets. On Communities and Crime it achieves an accuracy of 70.6% against the LR models and 56.0% against the MLP models compared to the 27.6% reached by the model-less attack. On Musk, it achieves 72.0% against the LR models and 41.7% against the MLP models compared to the 28.7% reached by the model-less attack.

6.6 Attribute inference attack

In this section, we present an attribute inference attack that uses as building blocks the correlations between input variables extracted from the model using our approach. We will refer to it as correlation inference-based attribute inference attack (CI-AIA).

Attack model. We assume the first variable in the dataset X_1 to be sensitive and consider an adversary that aims to recover the sensitive attribute value x_1 of a *target record* $((x_1, x_2, \ldots, x_{n-1}), y) \in D_T$ using black-box access to a model $\mathcal{M}_T$ trained on D_T. In line with the literature [127, 215, 167], we assume that the adversary knows the partial record $(x_2, \ldots, x_{n-1})$, the label y, and the one-way marginals of the attributes.

Attack methodology. Our intuition is that the correlation between the sensitive variable X_1 and the other variables, when combined with knowledge of the partial record $(x_2, \ldots, x_{n-1}, y)$, can lead to accurate inference of the sensitive attribute x_1. We propose an attack that extracts the correlations between the input variables $\rho(X_i, X_j), 1 \leq i < j \leq n-1$ from the model and uses them to generate synthetic data satisfying these correlations. We then search for (approximate) matches of the partial record $(x_2, \ldots, x_{n-1}, y)$ in the synthetic data and return as prediction $\hat{x_1}$ the average value of the sensitive attribute values of matched records.

Step 1: We execute our correlation inference attack independently against each target pair

$(X_i, X_j), 1 \leq i < j \leq n-1$ under default scenario **S2**. The output of the attack is a correlation bin $b_{i,j} \in \{1, \ldots, B\}$. To evaluate our attack on real-world datasets having arbitrary marginals, we run the attack using the ModifyCorrelationConstraints heuristic. We denote by $V'_{i,j} \in \mathbb{R}^{n-1}$ the modified correlation constraints returned by the heuristic.

Step 2: We sample synthetic datasets whose empirical correlations belong to the inferred bins, while also matching the correlation constraints $(\rho(X_i, Y))_{i=1,\ldots,n-1}$. Sampling synthetic data under constraints given by the adversary knowledge requires to modify the correlation constraints $\rho(X_i, Y), i = 1, \ldots, n-1$, as described in Sec. 6.3.5. We here use the *average* of modified constraints computed previously to separately attack each of the pairs:

$$V' = \frac{2}{(n-1)(n-2)} \sum_{1 \leq i < j \leq n-1} V'_{i,j} \tag{6.8}$$

We generate S' shadow datasets, then select among them the datasets D whose empirical correlations $C(D)_{i,j}$ belong to the inferred bin $b_{i,j}$, for every $1 \leq i < j \leq n-1$, discarding the rest. We concatenate the datasets together into a larger dataset D_{synth}.

Step 3: We select all the values of the sensitive attribute $x_1^1, \ldots, x_1^H$ belonging to records that approximately match the partial record $(x_2, \ldots, x_{n-1}, y)$. We search for approximate matches as follows. First, we discretize the marginals of non-sensitive variables $X_2, \ldots, X_{n-1}$ into G sub-intervals of same size and denote by $b_2, \ldots, b_{n-1}$ the size of the sub-intervals. Second, we retrieve all the records $(x'_1, \ldots, x'_{n-1}, y')$ in D_{synth} such that: (1) $|x_i - x'_i| \leq m_i b_i, i = 2, \ldots, n-1$ and (2) $y = y'$, where m_i denotes the resolution of the search around the partial record's attribute x_i. For instance, for $m_2 = 1$ and $b_2 = 0.2$ we only select records having $x'_2 \in [x_2 - 0.2, x_2 + 0.2]$. If no record is found, we increase the resolution parameters by increments δ_i: $m_i \leftarrow m_i + \delta_i, i = 2, \ldots, n-1$, until at least one record is found. We return the average sensitive attribute of matched records $\hat{x}_1 = \frac{1}{H}(\sum_{h=1}^{H} x_1^h)$ as our prediction.

Baselines. We compare our attack with five attacks from the literature and with two baselines. To perform attribute inference over a continuous sensitive attribute, we divide the attribute range into G bins of equal length, infer the most likely bin using a given attack, then return a uni-

form sample from the bin as the final prediction. Given a candidate record $(x_1^g, x_2, \ldots, x_{n-1}, y)$ where the value of the unknown sensitive attribute x_1 is substituted with a sample x_1^g of the g-th candidate bin, we will denote by V_y^g the confidence that the candidate record is classified with the target label y, i.e., the l-th element of output vector $\mathcal{M}_T(x_1^g, x_2, \ldots, x_{n-1})$.

1. **Fredrikson et al. [127]** The attack infers the most likely bin $g = 1, \ldots, G$ as per the probability $\Pr(x_1^g)\Pr(\hat{y} = y | \arg\max_{l=1}^{L} \mathcal{M}_T(x_1^g, x_2, \ldots, x_{n-1}) = \hat{y})$, where x_1^g is a value sampled uniformly at random in the g-th sub-interval. The second term is approximated using the confusion matrix of the target model, assumed by Fredrikson et al. [127] to be known by the attacker.

2. **CSMIA (Mehnaz et al. [215])** The attack sets the sensitive attribute value of the partial record to each possible value (here, one of G bins) and retrieves the model prediction $\hat{y}^g = \mathcal{M}_T(x_1^g, x_2, \ldots, x_{n-1}), g = 1, \ldots, G$. Note that we represent a bin using a uniform sample in the bin x_1^g. Then, the attack distinguishes three cases: (1) if $\hat{y}^g = y$ for only one value of g, the approach returns x_1^g, (2) if $\hat{y}^g = y$ for more than one value of g, the approach returns x_1^g on which the model is the most confident, and (3) if $\hat{y}^g \neq y$ for all values of g, the approach returns x_1^g on which the model is the least confident.

3. **Yeom et al. [354]** The attack returns the sensitive attribute value with the largest prior $\Pr(x_1^g)$ that passes a membership inference test. More specifically, the attack uses a membership oracle $\mathbb{O}$ that returns 1 if $(x_1^g, x_2, \ldots, x_{n-1}, y) \in D_T$ and 0 otherwise. The attack returns $\text{argmax}_g \Pr(x_1^g) \cdot \mathbb{O}(\mathcal{M}_T, (x_1^g, x_2, \ldots, x_{n-1}))$ (where x_1^g is sampled uniformly at random in the g-th bin). We use the threshold-based membership oracle of Yeom et al. [354], that returns 1 if and only if the model's confidence on the target record is larger than a threshold τ.

4. **Jayaraman and Evans [167]** removes the need for an oracle in the attack by Yeom et al. [354], proposing two variants. The first one is CAI, which returns $\arg\max_g V_y^g$. The second one is WCAI, which returns $\arg\max_g \Pr(x_1^g | x_2, \ldots, x_{n-1}) V_y^g$, making use of the conditional probability of the sensitive attribute given the non-sensitive attributes.

In their work, this quantity is estimated empirically on auxiliary data drawn from a similar distribution as the private dataset. As we here study a weaker adversary lacking access to auxiliary data, under our assumptions the attack would simply become $\arg\max_g \Pr(x_1^g)V_y^g$.

5. **Marginal prior.** This baseline returns a random sample from the one-way marginal of the sensitive attribute.

6. $X_1 - Y$ **correlation.** This baseline runs our CI-AIA approach on the variables X_1 and Y alone, with the goal to quantify the leakage from the adversary knowledge of $\rho(X_1, Y)$. Indeed, knowledge of the correlation $\rho(X_1, Y)$ can lead to better than random attribute inference of x_1 since y is known. This baseline acts as a lower bound for CI-AIA, as the latter additionally assumes access to the model. In practice, we run steps 2 and 3 of CI-AIA on variables X_1 and Y, but not step 1 as there is no need to extract additional correlations from the model.

Evaluation setup. We evaluate the attacks on the Fifa19 dataset, randomly selecting 1000 different data collections consisting of $n = 4$ columns. For each data collection, we train a Logistic Regression model and then execute each attack on the same, randomly selected, 500 records. We implement CI-AIA using the same parameters as in Sec. 6.4.3, $S' = 1000$ shadow datasets, $G = 100$ bins, and $m_i = 2$ and $\delta_i = 0.5$ for every $i = 1, \ldots, n-1$. We implement Yeom et al. [354] using $\tau = 0.5$.

Metric. To evaluate the accuracy of attacks, we divide the range of the sensitive attribute into three bins corresponding to the tertiles of the distribution. Thus, the marginal prior baseline attains an accuracy of roughly 33%.

Results. Table 6.2 shows CI-AIA to outperform previous works and baselines, achieving an accuracy of 49.8%, significantly better than CSMIA (Mehnaz et. al. [215]), which only reaches 46.5%, on par with CAI (Jayaraman and Evans [167]), and Fredrikson et al. [127], which reaches 38.4% and is on par with Yeom et al. [354]. Our attack also outperforms the $X_1 - Y$ correlation

Method	Accuracy
(Ours) CI-AIA	49.8 ± 1.0
Fredrikson et al. [127]	38.4 ± 0.8
CSMIA (Mehnaz et al. [215])	46.5 ± 0.9
Yeom et al. [354]	38.5 ± 0.8
CAI [167]	46.5 ± 0.9
WCAI [167]	41.7 ± 0.8
$X_1 - Y$ correlation	41.8 ± 0.9
Marginal prior	35.3 ± 0.6

Table 6.2: **Comparison between CI-AIA and other attribute inference attacks on Fifa19.** We report the attack accuracy averaged over 1000 runs (with 95% confidence interval).

baseline, confirming our intuition that the additional correlations $\rho(X_1, X_i), i > 1$ extracted from the model using our attack lead to more accurate attribute inference.

6.7 Related work

Property inference attacks (PIA), which we review in Sec. 5.3.2, study the leakage of macro properties of the dataset. Our work differs significantly from previous work in PIA. First, we focus, by design, on explicit micro characteristics of the dataset X rather than on external macro properties of the record (e.g., the gender label of an image). Second, our attack is weaker as it does not require access to an auxiliary dataset of real records following a similar distribution. Our shadow models are trained exclusively on synthetic records, making them accessible to weak adversaries. Third, previous works focus on a categorical choice between *very distinct* properties. For instance, Ganju et al. [131] aim to infer whether the proportion of older faces is equal to (precisely) 23% or 37% or if exactly 0% or 87% of the faces where from white people [131]. By contrast, we frame the correlation inference task as a prediction task *covering the entire range of possible values*. While Zhou et al. [368] aim to address this shortcoming by framing property inference as a regression task, they only apply their attack to Generative Adversarial Networks which would, by design, reflect the statistical properties of the training dataset making a correlation attack trivial.

Attribute inference attacks (AIA) aim to recover the sensitive attribute of a target record. Friedrikson et al. [127] proposed the first AIA against ML models. They applied their attack to linear regression models available as a black-box and aimed to infer a patient's genotype. Subsequent work [126] extended this attack to make use of confidence scores and applied it to decision trees. Yeom et al. [354] explored the connection between membership and attribute inference and proposed a black-box attack relying on a membership inference oracle. Mehnaz et al. [215] then proposed new black-box attacks: (1) a confidence-based attack dropping the prior work's assumption that the adversary knows the one-way marginals of the dataset and (2) a label-based attack assuming the adversary has access to a large fraction of the training dataset (excluding the sensitive attribute). In concurrent work, Jayaraman and Evans [167] propose two extensions of the black-box approach of Yeom et al. [354] and a white-box approach. We here study a weaker adversary lacking access to auxiliary data and propose an auxiliary data-free black-box AIA based on correlations extracted from the model.

Synthetic data-based shadow modelling is an under-explored research area. Shokri et al. [302] were the first to use synthetic data to train shadow models. They developed a hill-climbing algorithm to generate synthetic records. Their approach is based on the intuition that a model would be more confident on records that are similar to the training dataset. Salem et al. [283] argued that this method is only efficient for datasets of binary records and explored the use of shadow datasets from different sources. This is however not a suitable option when aiming to infer summary statistics of a dataset as it requires shadow datasets to span the entire range of possible values for the statistics.

6.8 Discussion

We here analyse some of the assumptions made on the adversary knowledge, then discuss whether correlation leakage is necessary for learning to happen.

Access to auxiliary data. Our approach does not require access to the private dataset D_T, nor to a dataset drawn from a similar distribution. Such detailed knowledge of the data is

a common assumption in the literature on inference attacks and has led to strong results as the adversary uncertainty is reduced, e.g., [131, 365, 52]. We however believe that real-world adversaries are unlikely to have detailed knowledge of the data. Our goal is thus to explicitly extract information about the data distribution and use it to infer attributes of individual records. While in this work we have assumed a Gaussian copula prior on the underlying distribution (allowing us to control for correlations), future work could explore different priors to improve our results.

Knowledge of the correlation constraints. Our default scenario **S2** assumes the adversary to know $(\rho(X_i, Y))_{i=1,\dots,n-1}$. We show that these can be extracted from the model using a small number of queries $\tilde{Q}$. More specifically, we sample $\tilde{Q}$ records $(x_1, ..., x_{n-1})$ independently from the marginals $F_1, \dots, F_{n-1}$, then query the model to retrieve the predicted label y. We then estimate $(\hat{\rho}(X_i, Y))_{i=1,\dots,n-1}$ on these records. We compute the $\mathbb{L}_1$ error of the predictions, defined as $\frac{1}{n}\sum_{i=1}^{n} |\hat{\rho}(X_i, Y) - \rho(X_i, Y)|$, over 500 data collections of $n = 4$ columns, and compare it with the $\mathbb{L}_1$ error of a random guess in $[-1, 1]$. Fig. 6.9 shows our approach to perform much better than the random guess, reducing the $\mathbb{L}_1$ error from 0.584 to 0.214 with as little as 100 queries to Logistic Regression models trained on the Fifa19 dataset. The performance of our approach improves as the number of queries $\tilde{Q}$ increases, but stabilizes early at $\tilde{Q} = 100$. Extending the correlation inference attack to the setting where the adversary does not know the correlation constraints is an interesting problem, which we leave for future work.

Knowledge of precise marginals. We have so far assumed the adversary to have precise knowledge of the marginals, discretising them for simplicity into $G = 100$ sub-intervals of equal size. Fig. 6.10 in the Appendix shows that decreasing G and, by doing so, reducing the adversary's ability to sample from the marginals with fine granularity, has a very small impact on the performance of our attack. The accuracy on Fifa19 decreases very slowly from 91.2% ($G = 100$) to 82.9% ($G = 5$), much higher than the model-less attack at 60.2%.

Knowledge of the model seed. We believe it more realistic to assume that the adversary does not know the seed used to initialise the training algorithm of MLP models, including their weights. This assumption leads to more uncertainty, as the shadow models may end up in

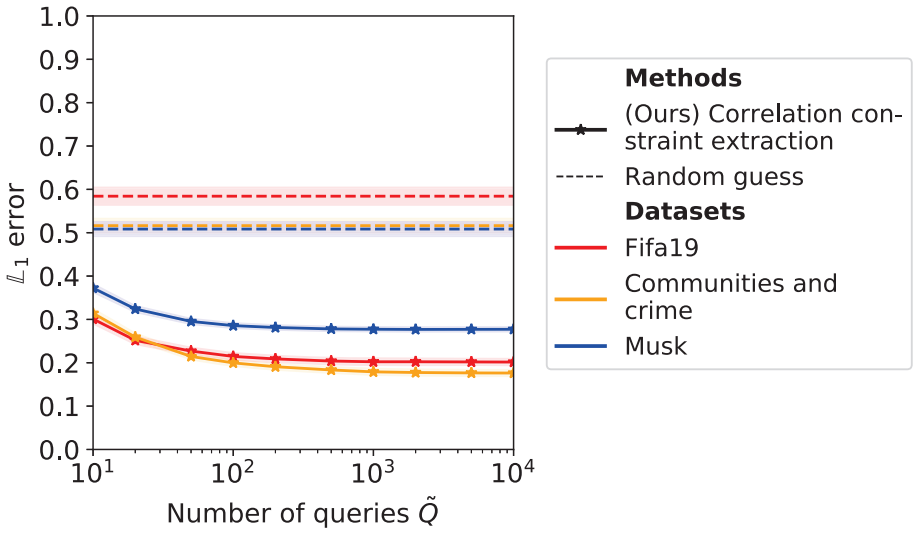


Figure 6.9: **Correlation constraint extraction from LR models for different number of queries $\tilde{Q}$.** We compute the $\mathbb{L}_1$ error between the inferred and the ground-truth values of the correlations: $\frac{1}{n}\sum_{i=1}^{n}|\hat{\rho}(X_i, Y) - \rho(X_i, Y)|$ (mean with 95% confidence interval)..

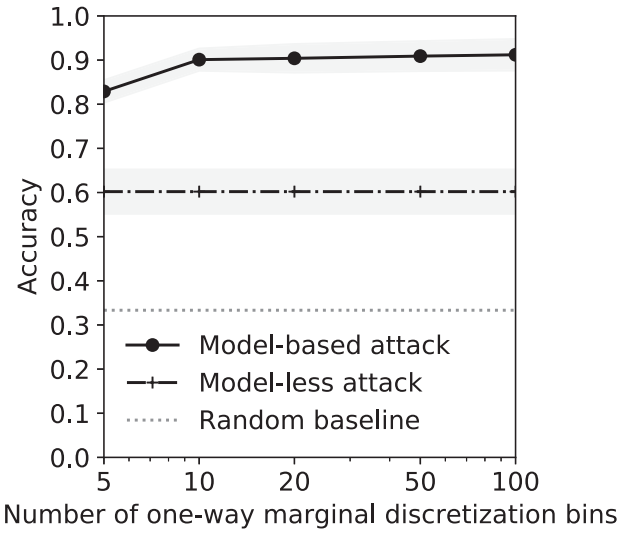


Figure 6.10: **Fifa19: Impact of the granularity of one-way marginals available to the attacker.** We report the accuracy (mean and standard deviation) of attacks on LR models as we vary the number of discretisation bins applied to the one-way marginals.

different local minima compared to the target model and may behave less similarly than expected. This is not an issue for LR models, as they minimise a convex objective having a global optimum [29]. Fig. 6.11A and B compare the performances of black-box and various white-box correlation inference attacks against MLP models, when changing, respectively keeping the seed constant between shadow and target models. While the black-box attack gains little from the seed knowledge (up to 4.6 p.p), the white-box attack reaches an astonishing 94.5% for $n = 3$

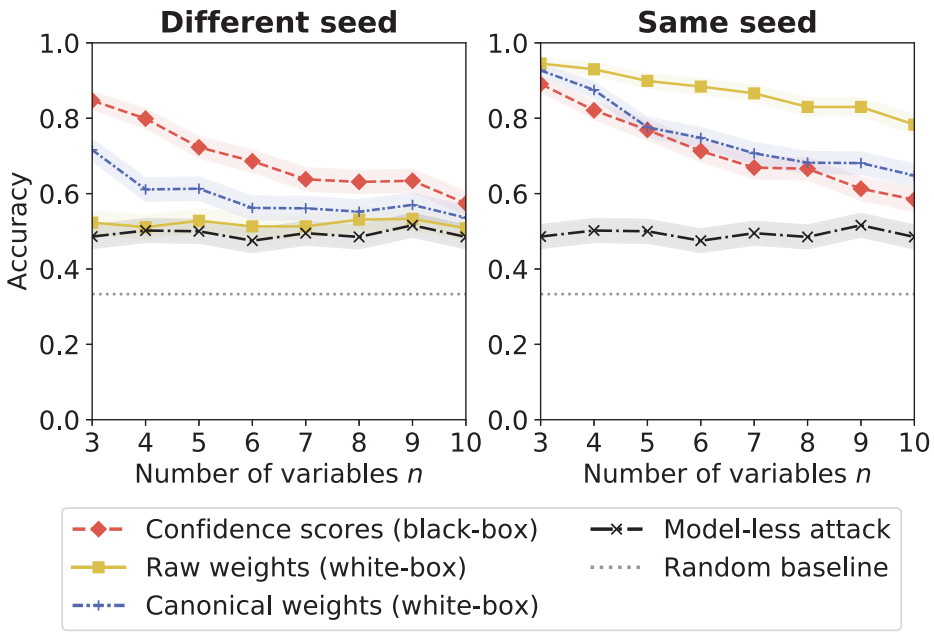


Figure 6.11: **Comparison between black-box and white-box CIAs against MLPs for different seed configurations between the shadow and target models**. We report the mean attack accuracy (with 95% confidence interval).

variables, 88.4 for $n = 6$, and 78.3% for $n = 10$ variables, the last being 29.8 p.p. above the model-less attack. It now also outperforms the attack on LR models for all $n \geq 4$, confirming our hypothesis that more weights (e.g. 432 for the MLP vs 10 for the LR when $n = 10$) can indeed encode more information. Finally, the canonical sorting of weights, proposed by Ganju et al. [131] to enhance the performance of the meta-classifier is, in this setting, somewhat effective when the seed changes, although still much lower than the black-box attack. It however achieves the opposite effect of destroying some of the information present in the raw weights when the seed is kept constant.

Potential reasons for the leakage. From a statistical theory perspective, our finding that models leak correlations of their input dataset raises questions. While the optimal model is only intended to learn $\Pr(Y|X = x)$, current learning techniques such as stochastic gradient descent optimise the expected loss computed over records of the joint distribution $\Pr(X, Y)$. Thus, the distribution of the input variables $\Pr(X)$ influences the model. Whether the leakage we observe is a necessary condition for learning to happen and thus, whether robust defences might exist, remains an open problem.

6.9 Conclusion

In this work, we study a new type of leakage in machine learning models, that of correlations between input variables of a tabular training dataset. We propose a novel *correlation inference attack*, developing two different approaches, *model-less* and *model-based*, allowing to correctly quantify the leakage from the model. We perform an extensive evaluation of our attacks, showing that models leak correlations of their training dataset with high accuracy, in both synthetic and real-world datasets. We also show that mitigations such as limiting the number of queries that can be made to the model, their precision, or training the models with differential privacy guarantees do not help. Finally, we show how correlations leaked from the model using our attack can be used as building blocks for attribute inference attacks.

Chapter 7

White-box membership inference attacks using re-aligned shadow models

The work presented in this chapter is based on the following, currently non peer-reviewed, manuscript: Cretu, A.-M., Jones, D., de Montjoye, Y.-A., and Tople, S. (2022). "Re-aligning Shadow Models can Improve White-box Membership Inference Attacks". A version of the manuscript is available at* `https://arxiv.org/pdf/2306.05093.pdf`.

7.1 Introduction

Membership inference risks have been extensively studied in the black-box setting, where the adversary accesses the model through an API [302, 283, 68, 52]. With models being increasingly deployed on-device (so-called *edge machine learning*), measuring the risk in the black-box setting could lead to underestimating the risks. Indeed, on-device *white-box* releases make more information available than black-box releases, i.e., the model architecture and weights. Beyond this, developing powerful attacks in the white-box setting is extremely important: once a model is released on edge devices, it cannot be taken back, and stronger attacks can be mounted at a later date.

Contrary to intuition, it has been shown that the *naïve* extension of shadow modelling to the white-box setting, by using features of internal layers (such as gradients, activations, or weights) in the provided order, leads to lower risks [229, 365, 80] than in the black-box setting. These results suggest that the white-box adversarial setting does not significantly increase the risk of information leakage, and thus that on-device deployment may be as safe as API releases.

Our contribution. In this work, we focus on multilayer perceptron and standard convolutional neural network (CNN) models. We show that the phenomenon of *misalignment* of learnt features in shadow models, where the different models learn similar features, but differently located in a layer [21, 199], is a key reason for the apparent paradox that white-box membership inference attacks have poor performance compared to the more restrictive black-box attacks. Fig. 7.1 illustrates the misalignment pheonomenon in CNN shadow models trained by an adversary. Misalignment has been studied in the ML fusion literature, in contexts where there is a need to combine multiple models, e.g., ensembling [21, 305], and has been used to interpret neural network representations [199] and the loss landscape of stochastic gradient descent [323, 113]. However, only a handful of works make this observation in the shadow modeling context [193, 131]. While they propose new attack techniques, they do not investigate its causes nor quantitatively measure its impact on membership inference attacks.

We here, first, systematically investigate the causes of misalignment in the context of shadow models. We observe misalignment to be a consequence of the uncertainty in the adversary knowledge while training shadow models to mimic the target model: uncertainty in the dataset, weight initialisation, ordering of mini-batches, and which neurons are being "dropped" as part of the dropout regularisation. We then show that while changing the weight initialisation results in misalignment changing the dataset or the randomness of the training algorithm does not lead to a noticeable difference.

Second, we extend and evaluate the effectiveness of re-alignment methods proposed in the model fusion literature to the shadow modelling context. We show these methods to successfully reduce the measured misalignment between the target and shadow models. Our results however suggest that re-alignment techniques are imperfect as they struggle with re-aligning large and

middle layers.

Finally, we perform a comprehensive evaluation of white-box membership inference attacks (MIA), analysing misalignment and the impact of subsequent re-alignment techniques on the effectiveness of white-box MIA attacks. We consider three different threat models. First, we study Nasr et al. [229]'s strong adversary who has access to part the private dataset which they use to extract perfectly aligned features from the target model. We argue that this adversary allows us to measure the gap between white-box and black-box attacks in the idealised setting of perfect alignment. Second, we study an intermediate adversary, informed by our analysis, that extracts features from shadow models trained using the same weight initialisation as the target model. Third, we study the classical adversary who trains misaligned shadow models using a different weight initialisation. Our analysis reveals that (1) MIAs suffer from misalignment between shadow models, but that (2) re-aligning the shadow models improves, sometimes significantly, MIA performance. On the CIFAR10 dataset with a false positive rate of 1%, white-box MIA using re-aligned shadow models improves the true positive rate by 4.5%. Taken together, our results highlight that on-device deployment increase the attack surface and that the newly available information can be used by an attacker.

7.2 Attack model

In this work, we instantiate the inference adversary described in Sec. 5.2 in the white-box membership setting. More specifically, we consider a target model $\mathcal{M}_T$ trained on a target, private dataset $D_T = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ and assume that the dataset is sampled from an underlying distribution $\mathcal{D}$. The adversary goal is to infer the membership of a target record $(x, y) \sim \mathcal{D}$, i.e., whether $(x, y) \in D_T$ using white-box access to the model.

Adversary uncertainty about the training algorithm. In line with the literature on inference attacks [23, 302, 229, 193, 365, 52, 318], we assume that the adversary has complete knowledge of the training algorithm $\mathcal{T}$. This means that the adversary can train from scratch a similar model on datasets of their choice. Importantly though, we do not assume the adversary

to have access to the same randomness (seed) that was used to train the target model. We believe our scenario to be much more likely than the alternative and stronger scenario of the attacker knowing the seed, for two reasons. First, model developers might not release the code to train a model, but only a description. Second, even when the code including the seed is available, ensuring reproducibility in neural network libraries is challenging due to non-determinism in the hardware [63]. Although rarely stated explicitly in previous works, this assumption is in line with how attacks are typically evaluated.

Adversary uncertainty about the private dataset. In line with the literature, to model the uncertainty of the adversary about the private dataset, we assume the adversary has access to an auxiliary dataset D_{A} sampled from the distribution $\mathcal{D}$ that is disjoint from the private dataset D_T. Like in prior work on white-box MIAs [193], we consider this threat model to be more realistic than the adversary having access to a large fraction of the target dataset and to an equal number of records known not to be part of the training dataset D_T [229] or to the entire dataset except for one target record [29]. We refer to our threat model as the *classical adversary*, and to the stronger adversary that has access to a large fraction of the dataset as the *auditor*.

7.3 Background

7.3.1 Deep neural networks

In this work, we focus on attacks against two broad classes of deep neural networks: multilayer perceptrons (MLP), consisting of fully connected layers, and standard convolutional neural networks (CNN), consisting of convolutional layers followed by fully connected layers. Both can be written as a function f consisting of a sequence of L layers: $f = g_L \circ \ldots \circ g_1$.

A fully connected (FC) layer applies an affine transformation, followed by a non-linear activation

function, to the input of the previous layer:

$$g_l(x^l) = \sigma_l(W^l x^{l-1} + b^l), \text{ with } \sigma_l : \mathbb{R}^{D^l} \to \mathbb{R}^{D^l} \text{ (non-linearity function)} \tag{7.1}$$

$$W^l \in \mathbb{R}^{D^l \times D^{l-1}} \text{ (weight matrix)} \tag{7.2}$$

$$b^l \in \mathbb{R}^{D^l} \text{ (bias vector)} \tag{7.3}$$

where $x^l \in D^l$ denotes the input to the l-th layer (and the output of the previous layer) for every $l = 1, \ldots, L$. For the input layer, we write $x^0 = x$. Popular choices of non-linear activation functions include ReLU (mapping each vector coordinate x_i to $\max(0, x_i)$), the *tanh*, and the *sigmoid*. The non-linear activation function used for the output classification layer ($l = L$) is the softmax: $\sigma(x_i^L) = e^{x_i^L} / \sum_{j=1}^{N_c} e^{x_j^L}$ for $i \in \{1, \ldots, N_c\}$, where N_c denotes the number of classes.

An FC layer can be represented through its *neurons*. More specifically, the l-th layer consists of D^l neurons, where the d-th neuron outputs a single activation value x_d^l. Its functionality can be described by:

1. A vector of *input weights* $(W_{d,1}^l, \ldots, W_{d,D_{l-1}}^l, b_d^l)$, consisting of the d-th row of the weight matrix and the corresponding bias term b_d^l. They are used to compute the output activation from the outputs of the previous layer: $x_d^l = \sigma_l(b_d^l + \sum_{d'=1}^{D^{l-1}} W_{d,d'}^l x_{d'}^{l-1})$.

2. A vector of *output weights* $(W_{d,1}^{l+1}, \ldots, W_{d,D^l}^{l+1})$. The d'-th weight is used to multiply the output activation x_d^l by the input weight of the d'-th neuron in the next layer.

In a convolutional layer, neurons are replaced with filters whose outputs are activation maps instead of single activation values. Let C^l be the number of filters in the l-th layer. Each filter $c \in \{1, \ldots, C_l\}$ is represented by a 3-D weight tensor $W^l[c]$ of size $C^{l-1} \times K_1^l \times K_2^l$ and a bias term $b_c^l \in \mathbb{R}$. Thus, the functionality of the layer can be represented by a 4-D weight tensor W^l and a 1-D vector b^l, concatenating the weight tensors and bias terms of individual filters, respectively.

Each filter acts independently on the input x^{l-1}, which is 3-D (e.g., an image) by applying a

convolution:

$$x^l[c] = b^l_c + \underbrace{\sum_{c'=1}^{C^{l-1}} W^l[c][c'] * x^{l-1}[c']}_{\bar{x}^l[c]} \tag{7.4}$$

The dimension of the input x^{k-1} is equal to $C^{l-1} \times Q^{l-1} \times R^{l-1}$, where C^{l-1} denotes the number of input channels, Q^{l-1} the image height, and R^{l-1} the image width. The dimension of the output x^l is equal to $C^l \times Q^l \times R^l$, where Q^l and R^l depend on Q^{l-1}, R^{l-1}, the sizes of the filters K^l_1 and K^l_2, as well as potentially other parameters (padding, dilation, stride). We refer the reader to Goodfellow et al. [140] or to library implementations [257] for the formulas used by the convolution operator $*$. The above equation expresses compactly that a scalar, the bias term b^l_c, is added to each element of the $Q^l \times R^l$ matrix $\bar{x}^l[c]$. The computation in Eq. 7.4 is typically followed by a non-linearity, pooling and dropout operations. When a convolutional layer g_l is followed by a fully connected layer, its output x^l is flattened to 1-D vector.

7.3.2 Symmetries in deep neural networks

Deep neural networks are known to exhibit symmetries [151, 21, 199, 358, 323, 305, 346, 113], which are essential to understanding the challenges of white-box attacks. Consider a deep neural network f_θ, with $\theta = ((W^1, b^1), \ldots, (W^L, b^L))$ its weight representation. A symmetry is any transformation T applied to the parameters θ that preserves the function: $\forall x, f_{T(\theta)}(x) = f_\theta(x)$. The main symmetries are:

1. Permutation equivalence. The positions of the neurons (filters) of internal layers can be arbitrarily permuted without changing f, as long as their corresponding input weights, bias, and output weights are permuted accordingly. For instance, in an FC layer g_l, applying a permutation to the rows of W^l, the bias vector b^l, and the columns of W^{l-1} does not change f.

2. Scale invariance. The input weights and the bias of a neuron can be multiplied by a constant $c > 0$ without changing f, so long as the output weights are multiplied by $1/c$.

This holds when there is no non-linearity, or when the non-linearity is the ReLU function.

3. Antisymmetry: The input weights, bias, and output weights of a neuron can be multiplied by -1 without changing f if the non-linearity is antisymmetric, e.g., $\forall x, \tanh(x) = -\tanh(-x)$.

The existence of symmetries implies that a neural network f_θ can be mapped to a large (potentially infinite) number of functionally identical networks $f_{\theta'}$ (i.e., $f_\theta(x) = f_{\theta'}(x), \forall x$) having different internal structures $\theta' \neq \theta$. Furthermore, it is well known that running a training algorithm $\mathcal{T}$ multiple times to optimise a non-convex loss function (typical with deep neural networks), leads the optimisation procedure to functionally different local minima (i.e., $\exists x : f_\theta(x) \neq f_{\theta'}(x)$). These two facts imply that models trained in different contexts, e.g., using different seeds or on different devices, can be functionally similar $f_\theta(x) \approx f_{\theta'}(x)$ for most x yet have very different weight representations θ and θ'. This is known as the *misalignment* phenomenon. This commonly manifests as a lack of reproducibility, which researchers and practitioners address by ensuring that the training algorithm is deterministic, i.e., fixing the seed and keeping track of the library and hardware used. Beyond this, misalignment is an issue when there is a need to combine models into a single one, since weights located in the same position may not have the same meaning across the models (see Sec. 7.4 for an overview).

7.3.3 Impact of misalignment on white-box shadow modelling

Shadow modelling, which we build upon in this work, is the main technique used to develop inference attacks against ML models (see Sec. 5.3 for a description). The core assumption underlying this approach is that the shadow models learn similar functions as the target model $\mathcal{M}_T$, so that the meta-classifier trained on features derived from shadow models performs well when applied to the target model. In a white-box membership inference setting, the features typically considered are the activations and gradients of internal layers of the model [229, 200]. These features are however likely to be misaligned between the shadow models and the target model, since they are trained using different datasets and randomness. This makes shadow

modelling harder, since the meta-classifier would have to learn a more difficult task: (1) the relationships between features and the membership of a record and (2) that this relationship is invariant to symmetries [131].

7.4 Related work

Misalignment has been studied in the model fusion literature where there is a need to train multiple models and then combine the models into a single one. Model fusion has e.g. been used to achieve the benefits of ensembling (better accuracy) without having to store multiple models [21, 305], to enable crossover mutations over neural networks in evolutionary algorithms [21], to average models in federated learning [358, 346]. Misalignment has also been used to better understand neural network representations [199] and the loss landscapes of stochastic gradient descent [323, 113]. Re-alignment techniques have been proposed, e.g., [21, 199], which we extend and apply for the first time to the shadow modelling context.

A handful of works have identified misalignment to be an issue for property inference attacks (PIA), a specific use of shadow models, and proposed solutions. Ganju et al. [131] looked at the permutation equivalence symmetry of MLP models, and showed it to affect the performance of the naïve PIA meta-classifier based on the model weights. The authors propose two approaches to tackle this challenge: weight-based neuron sorting and set-based meta-classifiers. Recently, Suri and Evans extended set-based meta-classifiers to CNNs [318]. Differently from these works, we systematically analyse the root causes of misalignment in shadow models and explore the effectiveness of re-alignment techniques. We then analyse the impact of misalignment and re-alignment techniques on white-box MIAs, the standard privacy test. More specifically, we show weight-based neuron sorting to be ineffective in reducing the misalignment and improving MIAs, and extend set-based classifiers to MIAs, showing them to perform worse or on par with re-alignment.

Impact of weight initialisation. Balle et al. [29]. developed an attack using shadow models to reconstruct a training record based on the model weights. They found that their attack

fails if the shadow models are not trained using the same weight initialisation as the target model. Similarly, Jagielski et al. [163] obtained tighter estimates of the privacy of the DP-SGD algorithm [8] when using a fixed weight initialisation. Our work provides an explanation for these empirical findings through the misalignment perspective and suggests re-alignment as a potential solution to improve attacks when the adversary does not know the weight initialisation.

Membership inference attacks have been studied extensively in the black-box setting, see e.g., [302, 283, 68, 332, 52, 200]. The features typically used are the model confidences on a record x with the main vulnerability exploited being that the model tends to be more confident on training records compared to unseen records. Together, these works have built a solid understanding of the conditions under which black-box MIAs are most likely to succeed, such as when the model is overfitted [302, 355], when the targeted records are outliers or members of subpopulations [205, 188], and when the auxiliary data comes from the same distribution and is more abundant [283, 332].

Comparatively very few works have developed white-box membership inference attacks. Nasr et al. [229] considered a strong adversary, which we refer to as *the auditor*, who knows a large number of member records $D_{\text{in}} \subset D_T$ and non-member records D_{out}, with $D_{\text{out}} \cap D_T = \emptyset$, and attempts to infer the membership of other records $x \sim \mathcal{D}$, with $x \notin D_{\text{in}} \cup D_{\text{out}}$. The authors train a meta-classifier (MC) on activations and gradients extracted *directly from the target model*. Shadow models are, however, not used in this setting, since the signal available through the target model is, by definition, perfectly aligned.

Leino and Fredrikson [194] considered the classical adversary who has access to a disjoint set of records $D_A \sim \mathcal{D}$, with $D_A \cap D_T = \emptyset$. The authors identify the issue of misalignment and address it by training *proxy* models that share the same internal representation as $\mathcal{M}_T$ in all the layers up to and including a given layer l. Their method is, however, computationally heavy. The proxy models are first: (1) initialised with the weights of the target model up to layer l and frozen during training and (2) initialised randomly in the remaining, upper layers. The function applied by the upper layers to each record is then linearly approximated using influence functions [194]. Finally, a meta-classifier is trained on a measure of behaviour

difference between the linear approximations of proxy and target models. The adversary needs to repeat this procedure by training shadow models to mimic the role of the target model and by training proxy models for each shadow model and layer. For computational reasons, we here instantiate their approach with the last layer and find that it performs worse than the naïve attack using the same features.

Sablayrolles et al. [281] aims to answer the question of whether the membership signal of internal layers is redundant with the model confidences. If they were completely redundant, white-box attacks could indeed never outperform black-box attacks. Under specific assumptions on the distribution of the model weights, they derive that the optimal attack only depends on the loss function. This implies that white-box attacks could not exploit additional membership signal to the optimal black-box attack, since the loss can be estimated from black-box outputs. These results however contradict empirical work that showed that white-box attacks *can* perform better than black-box, reporting double-digit accuracy gaps [200]. Further work is thus necessary to understand when the assumptions made by Sablayrolles et al. hold in practice.

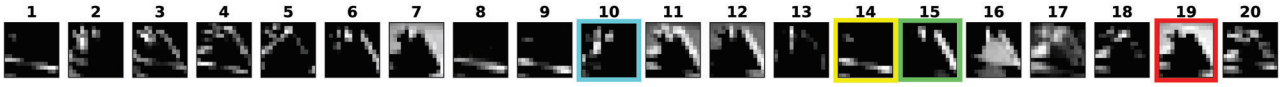
7.5 Problem statement

In this section, we show that misalignment occurs in CNN shadow models and demonstrate how it negatively impacts the performance of naïve white-box MIAs.

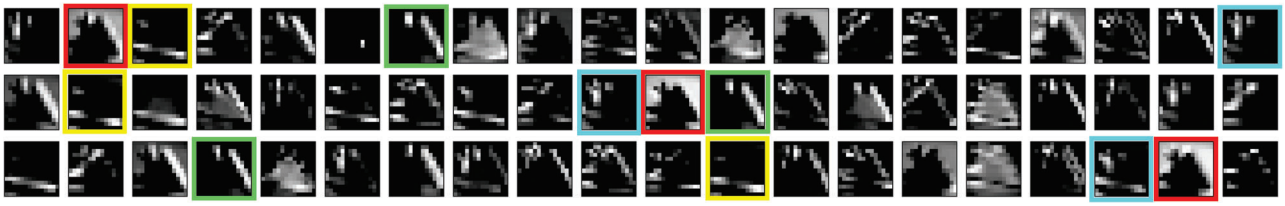
To illustrate the misalignment issue, we consider a standard CNN from previous work [193] which consists of two convolutional layers followed by two fully connected layers (see Sec. 7.6.3 for the complete details). We compare the internal representations of a target model $\mathcal{M}_T$ with those of shadow models trained by the *classical adversary*, who has no knowledge of the seed used to train the target model and uses a different seed for each shadow model. We randomly sample four mutually disjoint subsets of the CIFAR10 dataset [185] of size 12500, using the first subset to train a target model $\mathcal{M}_T$ and the others to train three shadow models. We visualise the activations maps, computed on an image of an aeroplane, of the first convolutional layer of each model. The activation maps describe the functionality of the convolutional layer. Each



(a) Image of an aeroplane



(b) Activations of the target model $\mathcal{M}_T$



(c) Activations of the shadow models trained by the classical adversary

Figure 7.1: **Visualisation of the misalignment problem in shadow models trained by the classical adversary.** Due to symmetries in the architecture, the models learn similar features in different positions. Each row shows the activations maps of the first layer of a CNN, taking as input the aeroplane image (top centre). The first row shows activations computed on the target model $\mathcal{M}_T$; the other rows show activations computed on three shadow models trained on disjoint datasets and using a different seed. The coloured rectangles highlight similar activation maps across different models.

activation map is a matrix of real values that are non-negative, due to the ReLU non-linearity function applied by the convolutional layer. We scale the values between 0 and 1 independently in each map, depicting 0 as black and 1 as white. The values indicate which parts of the image an activation map focuses on and which parts are "zeroed out" by the computation.

Fig. 7.1 shows that the target and shadow models learn many similar features (examples are highlighted in colour), but that these are not located in the same position consistently across models. This phenomenon also affects the upper, fully connected layers, although these are harder to visualise because their activations are scalars. Importantly, *misalignment does not affect the output classification layer*. Even though the outputs of shadow models might differ, they have the same functionality for target and shadow models: the l-th output consistently encodes the probability that the input image belongs to the l-th class. This explains why black-

box features extracted from shadow models perform very well, as reported in many works [302, 68, 365, 52].

In the context of white-box shadow modelling, the observed misalignment reduces the ability of the meta-classifier to effectively perform the inference task. We demonstrate this by first training a naïve white-box attack $\mathcal{A}_1$ to infer the membership of a record given as input the activations of the second to last layer. We train $\mathcal{A}_1$ on activations extracted from ten shadow models trained on shadow datasets sampled from the adversary knowledge D_{A} to predict the membership of records in the corresponding datasets. This attack achieves an Area Under the Curve (AUC) of 0.509, comparable to a random guess. Second, we train another attack $\mathcal{A}_2$ using as features the activations *computed directly on the target model*. To be able to do this, we assume the *auditor* threat model, having access to a large number of records of the distribution $\mathcal{D}$ together with their membership label (i.e., member or non-member). The auditor does not need to train shadow models [229] and does not suffer from the misalignment problem, since the attack features can be computed directly on the target model. This attack reaches a much higher AUC, of 0.805. The two experiments illustrate that the second to last layer contains signal useful for the membership task, but that this signal is not directly accessible from the shadow models.

7.6 Causes of shadow model misalignment

In this section, we identify the causes of misalignment in shadow models trained by the classical adversary. We consider all the factors in the adversary uncertainty that can introduce discrepancies between shadow models and the target model, and therefore influence the alignment of learnt features (Sec. 7.6.1). To understand the impact of individual factors, we change one factor at a time, then measure the resulting misalignment in each layer using three different metrics (Sec. 7.6.2). Then, we change multiple factors together, progressively increasing the adversary uncertainty, up to the classical adversary for which all selected factors are different between shadow and target models. From a security perspective, this means that we explore

intermediate attack scenarios which may lead to lower levels of misalignment than the one measured for the classical adversary (Sec. 7.6.3 and 7.6.4).

7.6.1 Selected factors

Shadow models typically differ from the target model in the following factors:

(1) **Training dataset.** A shadow model is trained on a different dataset compared to the target model, otherwise the adversary would already know everything about the target dataset. The training dataset of a shadow model can be partially overlapping with D_T [52] or disjoint from D_T [302, 131].

(2) **Model initialisation**. This is the value of the model weights at the beginning of training. The model weights are typically sampled randomly and independently from a distribution centred around zero, e.g., uniform or normal with parameters dependent on the layer sizes [137, 257].

(3) **Batch ordering**. This is the ordering of mini-batches on which the gradient is estimated in each training step. At the beginning of each epoch (representing a complete pass over the dataset), the training dataset is randomly shuffled. Consequently, the use of different training randomness between the shadow models and the target model leads to different batch ordering.

(4) **Dropout selection**. Dropout [310] is a popular regularisation technique aiming to reduce overfitting. Dropout is typically applied to the outputs of the internal layers. In each forward pass, dropout randomly sets each value of an input vector independently to 0 with probability p, or multiplies it by $\frac{1}{1-p}$ with probability $1-p$. Consequently, the use of different training randomness between the shadow models and the target model leads to different values being "zeroed out".

While this list is exhaustive for the neural networks considered in this work, additional sources of randomness could be considered, e.g., gradient perturbation in the case of the DP-SGD

algorithm [8] commonly used to train models with differentially private guarantees.

The standard way to control the randomness of factors (2)-(4) is to use a common pseudorandom number generator (PRNG), setting its seed at the beginning of training. To measure the influence of each factor, we use a separate PRNG for each factor, seeding it separately. We train a target model $\mathcal{M}_T$ on a target dataset D_T and a random choice of different seeds for each factor. Then, given one or more factors of interest whose influence we want to measure, we train multiple shadow models. The shadow models are trained in the same way as the target model, except for the dataset (if the dataset is included in the factors of interest) and the seeds corresponding to the factors of interest, which we set to different random values.

7.6.2 Metrics to measure the misalignment

Our goal is to measure, in each layer, how dissimilar the neurons of the shadow model are compared to those of target model. We represent the functionality of a neuron either by its input weights or its output activation (a scalar for FC layers and a 2-D activation map for convolutional layers). Given a shadow model $\mathcal{M}$ and the target model $\mathcal{M}_T$, we compute the following per-layer misalignment scores:

1. Weight misalignment score (WMS). In a layer l, this metric is defined as

$$\sqrt{||W^l(\mathcal{M}_T) - W^l(\mathcal{M})||_2^2 + ||b^l(\mathcal{M}_T) - b^l(\mathcal{M})||_2^2} \tag{7.5}$$

i.e., the Euclidean distance between weights (and biases) of the same layer in the target and shadow model. Here, we denote by $W^l(\mathcal{M})$ the weight matrix of the l-th layer of $\mathcal{M}$ (and by $b^l(\mathcal{M})$ its bias vector). In a convolutional layer, the metric is defined analogously over the weights and bias of individual filters.

2. Activation misalignment scores (AMS): For a layer l and a record x, the activation misalignment score is defined as $||x^l(\mathcal{M}_T) - x^l(\mathcal{M})||_2$, i.e., the Euclidean distance between the output activations computed on the target model and shadow model, respectively. For an FC

layer, x^l is a vector of size D^l. For a convolutional layer, x^l is a 3-D tensor, consisting of C^l activation maps, that we flatten. We compute the average AMS over R records.

3. Correlation between activations. We complement the two metrics with a third one [199], which measures how correlated neurons in the same position are across two models. In an FC layer l and a neuron d, consider the random variable x_d^l, where the randomness is taken over input records $x \sim D_{\mathrm{A}}$. Passing R records through a model gives a series of R activation values $S_d := ((x_1)_d^l, \ldots, (x_R)_d^l)$. Then, the metric computes the average (over all the positions d) of the Pearson correlation coefficient between the two series $S_d(\mathcal{M}_T)$ and $S_d(\mathcal{M})$. In a convolutional layer, activation maps replace the activation values. We randomly select P pixels among the $Q^l \times R^l$ available in the activation map, once before computing the metric. To measure the similarity between two filters, we compute, as before, the correlation between the time series of pixel p, then average it out over the P pixels. Finally, the metric computes the average similarity over all positions $c = 1, \ldots, C^l$.

We select these metrics for their interpretability, and to cover the functionality of a neuron (filter) in terms of both weights and output activations. The first two scores range from 0 to ∞, are equal to 0 for two identical models, with lower values indicating a larger similarity between the internal representations of the models at layer l. The third score ranges between -1 and 1. A value close to 0 indicates low correlation between the internal representations, while close to 1 shows high similarity. Additionally, the correlation is scale-invariant, which allows to compare different layers.

Random permutation. As a baseline, in each layer, we compute our metrics on the target model and the same model where a random permutation is applied to the neurons (filters) of the layer. This baseline allows us to contextualise the change incurred by the selected factors of difference between a shadow and a target model.

7.6.3 Experimental setup

Dataset. We perform experiments on the CIFAR10 dataset [186], which consists of 60000 colour images of size 32x32, labeled with one of 10 classes. The dataset is made available as a training split of 50000 and a test split of 10000 images. We randomly sample 12500 records from the train split as our target dataset D_T. For experiments using shadow models trained on datasets disjoint from D_T, we randomly sample from the train split $N = 3$ other datasets of same size, disjoint from one another and from D_T. For experiments using shadow models trained on overlapping datasets, we randomly sample $N = 5$ datasets of 12500 records from the training split. Consequently, the average dataset overlap between two different models is of 25%.

Model architecture and training details. We use a 4-layer CNN consisting of 2 convolutional layers and 2 fully connected layers. The first convolutional layer has $C^{l=0} = 3$ input channels and $C^{l=1} = 20$ output channels, filter sizes $K_1^1 = K_2^1 = 5$, followed by the ReLU non-linearity, a 2×2 max pooling layer, and a dropout layer with $p = 20\%$. The second layer uses the same parameters except that the number of output channels is equal to $C^{l=2} = 50$. The third layer is fully connected and maps the input of size $D^{l=2} = 1250$ to an output of size $D^{l=3} = 500$, followed by ReLU non-linearity and a dropout layer with $p = 20\%$. The final, fourth layer is fully connected and maps the input of size $D^{l=3} = 500$ to an output of size $D^{l=4} = 10$. We train the network using the Adam optimiser [180] using a batch size of 64, a learning rate of $\eta = 0.01$, dividing it by 2 after $E = 5$ epochs of non-increasing accuracy on a validation set of 5000 records from the test split. We refer to E as the early stopping patience. We stop the training when η drops below 10^{-5} or after 100 epochs, whichever comes first. The architecture and training hyperparameters are the same as in prior work [193], except that we use a smaller learning rate that gives us better utility and early stopping to prevent overfitting.

Metric parameters. We compute the activation misalignment score on $R = 500$ validation records (the same for all the models). We compute the correlation score on the same records and $P = 50$ pixels.

7.6.4 Results

Table 7.1 shows that **the main cause for the misalignment of shadow models trained by the classical adversary is their different model initialisation**. More specifically, when shadow models are trained on the same dataset, with the same batch ordering and dropout selection as the target model, but with a *different model initialisation*, the weight misalignment scores reach values comparable to applying a random permutation (RP) to the neurons (filters) of the target model $\mathcal{M}_T$. In the first layer, the score computed over shadow models having a different model initialisation ($\neq$ MI) is 12.09, similar to applying a random permutation to the filters of the target model (11.81). In contrast, shadow models differing only in their batch ordering ($\neq$ BO) or dropout selection ($\neq$ DS) are aligned to the target model, reaching scores of 3.42 and 3.52, respectively.

Interestingly, shadow models differing from the target model in their training dataset (but not in the other factors) are also broadly aligned to the target model, reaching a weight misalignment score of 4.32 and 5.11 when trained on a dataset that can be overlapping with, respectively disjoint from, D_T. Fig. 7.2 and 7.3 illustrate that the activation maps of the first convolutional layer are indeed aligned to those of the target model when changing only the batch ordering, dropout selection, or the training dataset. Our visualisations allow to directly compare these shadow models with the ones having a different model initialisation.

We also instantiate concrete attack scenarios by changing multiple factors together. More specifically, Table 7.1A shows that changing everything but the model initialisation ($\neq$ BO, $\neq$ DS, and disjoint dataset) results in a score of 5.17, much lower than the classical adversary scenario, which additionally assumes a different model initialisation (12.44). **From a security perspective, this strongly suggests that adversaries having knowledge of the model initialisation can train shadow models which, internally, are much more similar to the target model than the classical adversary.** This assumption holds whenever the seed is available to the adversary, or when the target model is fine-tuned and its pretrained weights are available, or when the adversary accesses a target model before a mini-batch update is performed [282, 164]. Fig. 7.4 shows that the activation maps of these shadow models are

overall well aligned to the target model. However, since now multiple factors are different, we also observe some differences, notably in columns 1 and 7.

The trends identified in the first layer also hold for the other layers. However, we observe smaller gaps between $\neq$ MI compared to e.g., $\neq$ BO. More specifically, the ratio between the corresponding scores decreases from $12.09/3.42 \approx 3.4\times$ in the first layer to $1.9\times$ in the second layer, reaching a minimum of $1.4\times$ in the third layer. This suggests that in some of the layers, even the smallest change can result in large differences between weights compared to the target model.

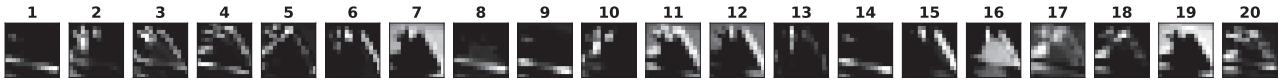
Impact of hyperparameters. To understand the impact of the training hyperparameters on the weight misalignment scores, we use different hyperparameters for the target model (propagating them to the shadow model). More specifically, we study the impact of the learning rate η, as it is proportional to changes applied to the weights in each gradient update, and on the early stopping patience E, as larger values can lead to longer training and therefore increase model overfitting. Table 7.3 shows that although numbers vary, the trends of weight misalignment scores are similar to the ones identified previously. To minimise the impact of overfitting, in this experiment we trained the models on 50000 records, the largest training dataset size considered in the work. Note that changing the dataset sizes impacts the magnitude of the scores. For completeness, we report in Table 7.2 the weight misalignment scores for various training dataset sizes and illustrate in Fig. 7.5 the activation maps in the first convolutional layer.

Other metrics. Table 7.1B shows the output activations to follow the same trends as the weights. There is one exception: the fourth layer, whose output activations are always aligned, as they are trained to encode per-class probabilities. For this reason, any difference between shadow models and $\mathcal{M}_T$ yields a much lower score compared to randomly permuting this layer. The results on the fourth layer suggest that the input-output behaviours of shadow models deviate from the target model more when using $\neq$ MI (0.32) or a disjoint dataset (0.41) compared to applying other changes, e.g., $\neq$ BO (0.28). We conclude that shadow models always learn a different classifier compared to the target model.

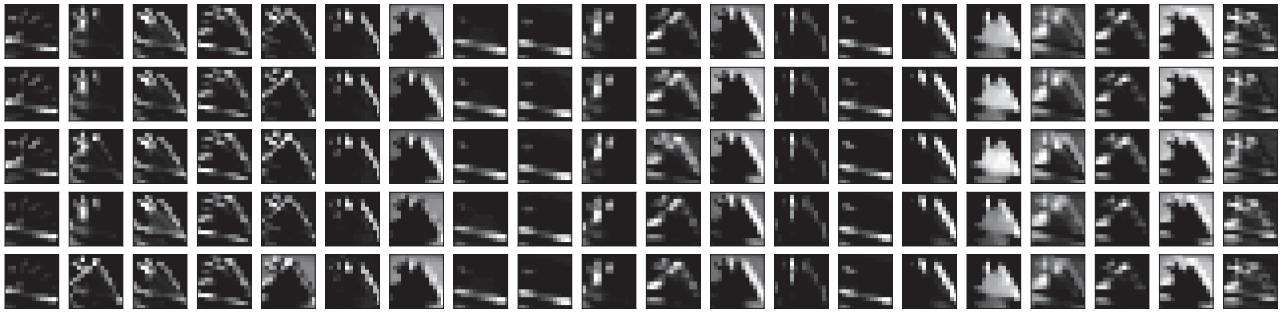
Table 7.1C reports similar trends, with the activation correlation scores additionally allowing to compare the levels of misalignment in different layers. As a reminder, we report the correlation between activations of neurons (filters) located in same position across shadow and target models. We again observe that changing the model initialisation has a very large impact on the correlation, across all internal layers. This result explains the low AUC of the MIA reported in Sec. 7.5: the outputs of the third (second to last) layer in shadow models are uncorrelated with the target model ($\rho = 0.01$, All $\neq$). The positive correlation between the first layer and its permuted self ($\rho = 0.15$), suggests the existence of redundancies in features learnt in the first layer. For all the factors considered, the correlation decreases as we move from lower to upper layers.

Table 7.1: **Misalignment scores using different metrics** (mean and standard deviation). Model initialisation is the most influential individual factor responsible for misalignment. The classical adversary corresponds to All $\neq$, while the adversary having knowledge of the target model initialisation corresponds to $\neq$ BO, $\neq$ DS, and DD (both are highlighted in **bold**).

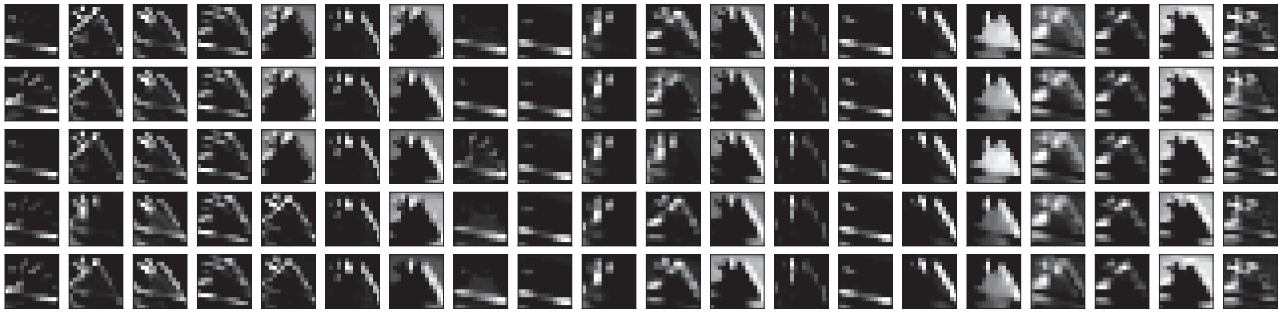
Difference w.r.t. target model $\mathcal{M}_T$	**First layer (20 filters)**	**Second layer (50 filters)**	**Third layer (500 neurons)**	**Fourth layer (10 neurons)**
A. Weight misalignment scores (↓ indicates better alignment)				
Random permutation	11.81 (0.51)	16.58 (0.15)	31.87 (0.06)	12.80 (0.66)
$\neq$ Model initialisation (MI)	12.09 (0.34)	16.24 (0.20)	30.46 (0.45)	12.54 (0.14)
$\neq$ Batch ordering (BO)	3.42 (0.34)	8.88 (0.26)	21.51 (0.85)	6.62 (0.15)
$\neq$ Dropout selection (DS)	3.52 (0.30)	8.72 (0.23)	21.25 (0.46)	6.47 (0.16)
Overlapping datasets	4.32 (0.36)	10.52 (0.31)	23.20 (0.60)	7.23 (0.11)
Disjoint datasets (DD)	5.11 (0.59)	11.21 (0.11)	24.24 (0.35)	7.56 (0.05)
$\neq$ BO, $\neq$ DS, and DD	5.17 (0.31)	11.46 (0.03)	24.92 (0.21)	7.55 (0.07)
$\neq$ MI, $\neq$ BO, and $\neq$ DS	12.05 (0.43)	16.30 (0.21)	30.53 (0.34)	12.65 (0.18)
All $\neq$ ($\neq$ MI, $\neq$ BO, $\neq$ DS, and DD)	12.44 (0.36)	16.56 (0.20)	30.98 (0.50)	12.87 (0.28)
B. Activation misalignment scores (↓ indicates better alignment)				
Random permutation	68.59 (3.10)	72.70 (0.65)	28.60 (0.05)	1.02 (0.12)
$\neq$ Model initialisation (MI)	68.31 (1.96)	70.99 (1.46)	27.34 (0.53)	0.32 (0.01)
$\neq$ Batch ordering (BO)	20.87 (2.21)	37.37 (1.27)	22.35 (0.50)	0.28 (0.01)
$\neq$ Dropout selection (DS)	21.88 (2.45)	36.13 (1.01)	21.88 (0.52)	0.27 (0.01)
Overlapping datasets	25.28 (2.06)	44.51 (0.67)	23.73 (0.46)	0.39 (0.01)
Disjoint datasets (DD)	28.41 (2.79)	46.49 (0.49)	24.66 (0.35)	0.41 (0.01)
$\neq$ BO, $\neq$ DS, and DD	30.54 (4.15)	47.73 (1.40)	24.85 (0.29)	0.41 (0.02)
$\neq$ MI, $\neq$ BO, and $\neq$ DS	68.90 (1.99)	71.45 (1.72)	27.59 (0.70)	0.32 (0.01)
All $\neq$ ($\neq$MI, $\neq$ BO, $\neq$ DS, and DD)	68.64 (1.93)	72.13 (0.29)	27.90 (0.45)	0.42 (0.03)
C. Activation correlation scores (↑ indicates better alignment)				
Random permutation	0.15 (0.07)	0.04 (0.02)	0.01 (0.00)	0.04 (0.05)
$\neq$ Model initialisation (MI)	0.13 (0.03)	0.03 (0.01)	0.01 (0.00)	0.28 (0.07)
$\neq$ Batch ordering (BO)	0.74 (0.06)	0.67 (0.02)	0.28 (0.02)	0.29 (0.07)
$\neq$ Dropout selection (DS)	0.73 (0.04)	0.68 (0.01)	0.29 (0.01)	0.29 (0.07)
Overlapping datasets	0.71 (0.06)	0.56 (0.02)	0.20 (0.02)	0.26 (0.06)
Disjoint datasets (DD)	0.71 (0.06)	0.55 (0.02)	0.18 (0.01)	0.28 (0.07)
$\neq$ BO, $\neq$ DS, and $\neq$ DD	0.70 (0.05)	0.53 (0.02)	0.18 (0.01)	0.28 (0.08)
$\neq$ MI, $\neq$ BO, and $\neq$ DS	0.13 (0.03)	0.04 (0.01)	0.01 (0.00)	0.28 (0.07)
All $\neq$ ($\neq$MI, $\neq$ BO, $\neq$ DS, and DD)	0.14 (0.03)	0.02 (0.01)	0.01 (0.00)	0.28 (0.08)



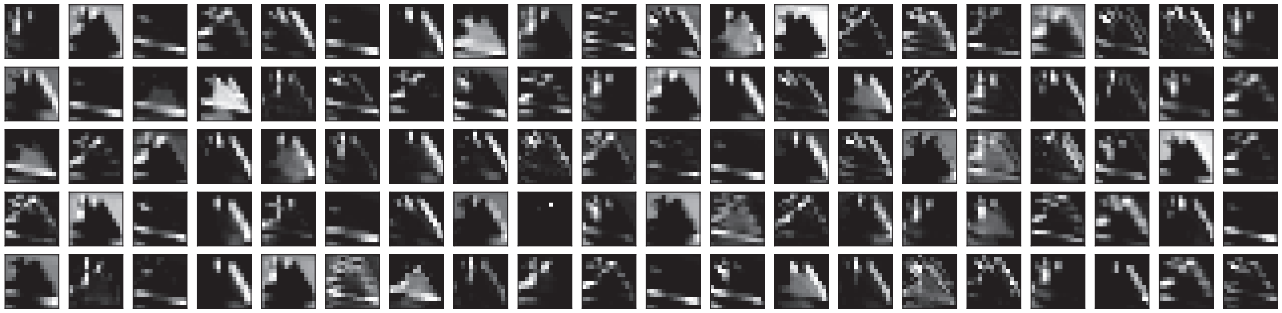
(a) Target model $\mathcal{M}_T$



(b) Different batch ordering

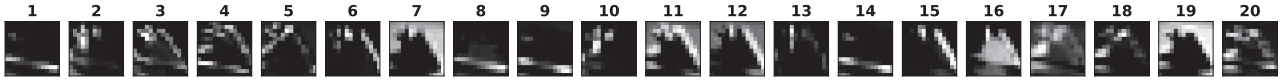


(c) Different dropout selection

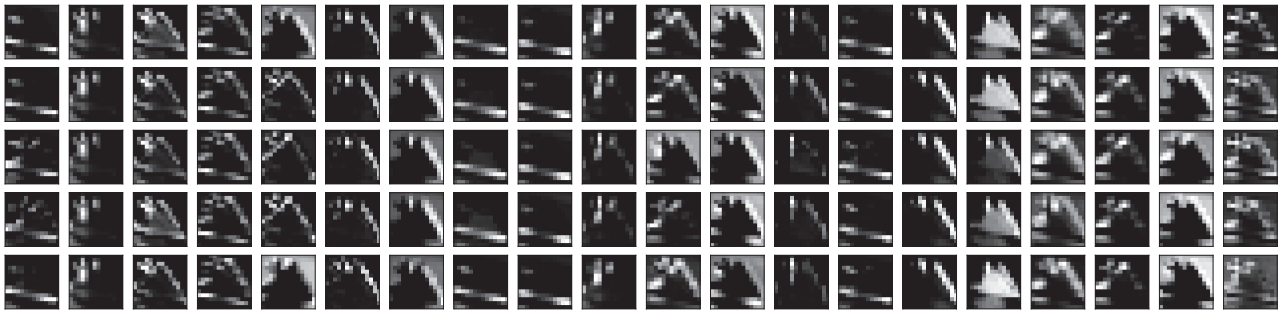


(d) Different model initialisation

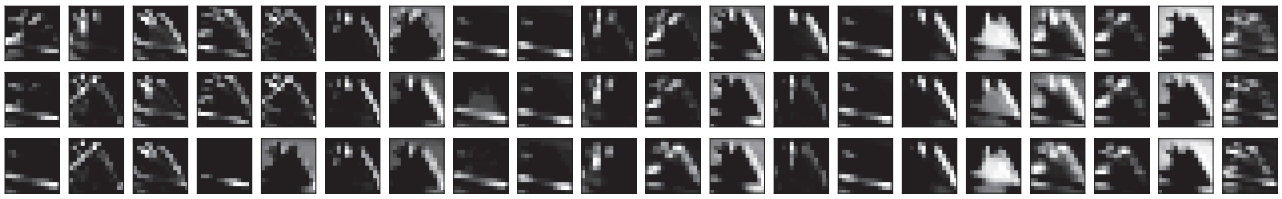
Figure 7.2: **Visualisation of the first layer of CNN shadow models** when changing only the seed of (b) the batch ordering, or (c) the dropout selection, or (d) the model initialisation with respect to the target model. Changing the seed of the model results in layer misalignment between the target model $\mathcal{M}_T$ and the shadow models. Changing the seeds controlling the other sources of randomness does not, although some of the feature maps are slightly different.



(a) Target model $\mathcal{M}^T$

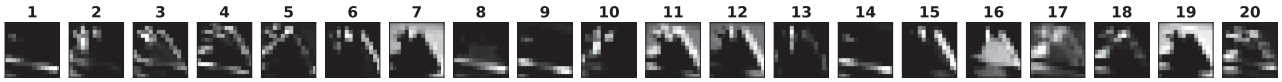


(b) Overlapping datasets

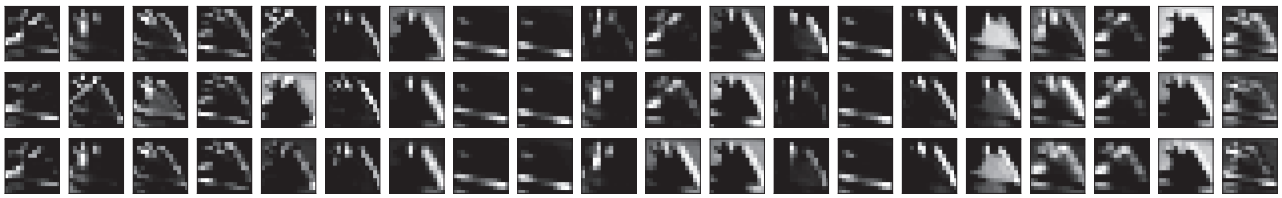


(c) Disjoint datasets

Figure 7.3: **Visualisation of the first layer of CNN shadow models** trained on a dataset that is (b) overlapping with or (c) disjoint from the training dataset of the target model. Changing the dataset slightly impact the activation maps but does not result in misalignment, even when using disjoint datasets.



(a) Target model $\mathcal{M}_T$



(b) Shadow models trained using the same model initialisation

Figure 7.4: **Visualisation of the first layer of CNN shadow models having the same weight initialisation as the target model.** The shadow models are trained using a different batch ordering, dropout selection, and a dataset disjoint from the target model. The activation maps are broadly aligned although we can see some differences, e.g., in columns 1 and 7.

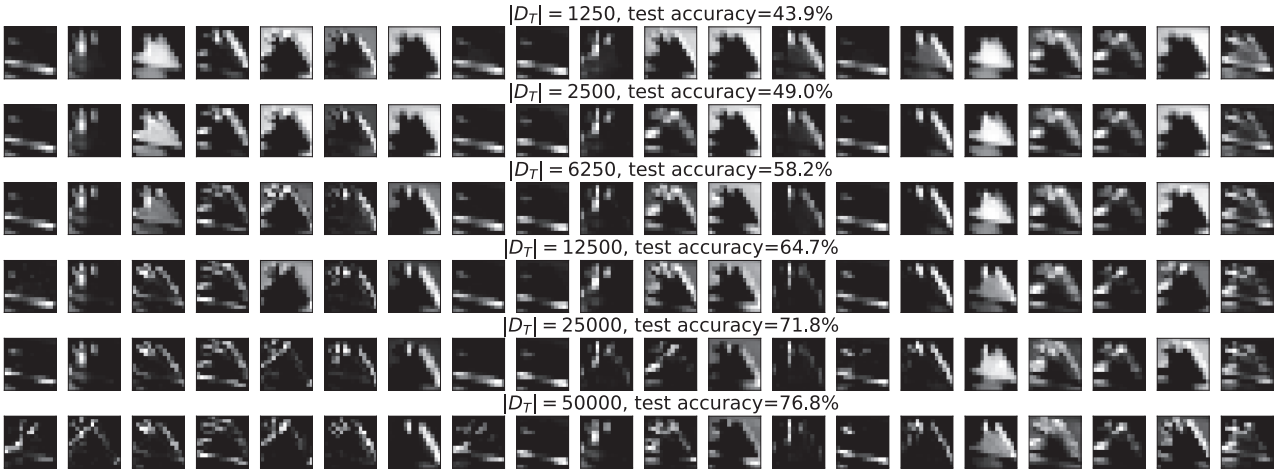


Figure 7.5: **Visualisation of the first layer of CNN shadow models trained on datasets of different sizes**. Each dataset is a subset of the larger datasets. Increasing the size of the dataset gradually changes the activation maps. As the model becomes less overfitted and learns more robust features, the activation maps stabilise and remain aligned.

Table 7.2: **Weight misalignment scores** for different training dataset sizes. In all of the settings, changing the model initialisation leads to misalignment.

Training size	**Difference w.r.t. target model $\mathcal{M}_T$**	**First layer 20 filters**	**Second layer 50 filters**	**Third layer 500 neurons**	**Fourth layer 10 neurons**
12500	Random permutation	11.81 (0.51)	16.58 (0.15)	31.87 (0.06)	12.80 (0.66)
	$\neq$ Model initialisation (MI)	12.09 (0.34)	16.24 (0.20)	30.46 (0.45)	12.54 (0.14)
	$\neq$ Batch ordering (BO)	3.42 (0.34)	8.88 (0.26)	21.51 (0.85)	6.62 (0.15)
	$\neq$ Dropout selection (DS)	3.52 (0.30)	8.72 (0.23)	21.25 (0.46)	6.47 (0.16)
	$\neq$ MI, $\neq$ BO, and $\neq$ DS	12.05 (0.43)	16.30 (0.21)	30.53 (0.34)	12.65 (0.18)
25000	Random permutation	13.94 (0.58)	19.08 (0.29)	35.49 (0.06)	14.14 (0.74)
	$\neq$ Model initialisation (MI)	14.58 (0.32)	19.56 (0.51)	36.02 (1.30)	14.25 (0.16)
	$\neq$ Batch ordering (BO)	5.62 (1.89)	12.27 (0.53)	28.66 (1.03)	7.87 (0.15)
	$\neq$ Dropout selection (DS)	5.63 (0.12)	12.05 (0.23)	28.64 (0.39)	7.80 (0.11)
	$\neq$ MI, $\neq$ BO, and $\neq$ DS	14.48 (0.32)	19.39 (0.22)	35.63 (0.31)	14.31 (0.10)
50000	Random permutation	16.74 (0.52)	24.50 (0.41)	48.78 (0.07)	15.60 (0.87)
	$\neq$ Model initialisation (MI)	17.23 (0.19)	24.23 (0.19)	47.19 (0.45)	15.51 (0.13)
	$\neq$ Batch ordering (BO)	8.01 (0.77)	16.66 (0.61)	41.41 (1.60)	9.29 (0.08)
	$\neq$ Dropout selection (DS)	6.54 (1.25)	15.41 (0.38)	40.51 (0.71)	9.19 (0.06)
	$\neq$ MI, $\neq$ BO, and $\neq$ DS	17.33 (0.29)	24.53 (0.37)	48.09 (1.02)	15.75 (0.20)

Table 7.3: **Weight misalignment scores** for different values of the learning rate η and the early stopping patience E. The training dataset size is 50000. Regardless of the setting, changing the model initialisation leads to misalignment of internal layers that is comparable to applying a random permutation to the layer.

	Difference w.r.t. target model $\mathcal{M}_T$	**First layer 20 filters**	**Second layer 50 filters**	**Third layer 500 neurons**	**Fourth layer 10 neurons**
$\eta = 0.01, P = 5$	Random permutation	16.74 (0.52)	24.50 (0.41)	48.78 (0.07)	15.60 (0.87)
	$\neq$ Model initialisation	17.23 (0.19)	24.23 (0.19)	47.19 (0.45)	15.51 (0.13)
	$\neq$ Batch ordering	8.01 (0.77)	16.66 (0.61)	41.41 (1.60)	9.29 (0.08)
$\eta = 0.001, P = 5$	Random permutation	9.37 (0.34)	11.19 (0.14)	21.25 (0.03)	9.31 (0.47)
	$\neq$ Model initialisation	9.86 (0.28)	11.38 (0.15)	21.31 (0.08)	9.48 (0.13)
	$\neq$ Batch ordering	1.02 (0.10)	2.62 (0.08)	5.97 (0.20)	2.26 (0.05)
$\eta = 0.001, P = 10$	Random permutation	10.27 (0.36)	12.13 (0.14)	22.04 (0.03)	10.27 (0.51)
	$\neq$ Model initialisation	10.96 (0.29)	12.50 (0.14)	22.25 (0.08)	10.63 (0.07)
	$\neq$ Batch ordering	1.24 (0.12)	3.16 (0.09)	7.75 (0.12)	2.75 (0.07)

7.7 Re-aligning the layers

Our findings suggest that permutation equivalence is the main symmetry responsible for the misalignment between shadow models trained by the classical adversary. Indeed, the misalignment measured in these models is comparable to applying a random permutation to internal layers of $\mathcal{M}_T$. A natural question is whether the internal layers of shadow models can be re-aligned to those of the target model, by seeking the "right" way to permute the neurons (filters) so as to minimise misalignment. While there may not be a one-to-one mapping between the neurons, i.e., some neurons in a shadow model may be functionally different from all the neurons in the target model, correctly re-aligning even a subset of neurons may significantly reduce misalignment and improve MIA performance.

7.7.1 Our approach

We here evaluate several re-alignment techniques from the ML literature [21, 199] that are instantiations of the bipartite assignment problem. More specifically, we keep the target model $\mathcal{M}_T$ fixed and search for a permutation σ_l to neurons (filters) in the l-th layer of the shadow model $\mathcal{M}$ so as to maximise its alignment to the l-th layer of $\mathcal{M}_T$. If $n_1, \ldots, n_{D_l}$ are the neurons of $\mathcal{M}$ and $n_1^T, \ldots, n_{D_l}^T$ are the neurons of $\mathcal{M}_T$, we seek the optimal permutation of neurons of the shadow model $\sigma_l(n_1), \ldots, \sigma_l(n_{D_l})$, i.e., that maximises the alignment between the neurons: $\sigma_l = \arg\max_\sigma \sum_{1 \leq d \leq D_l} sim(\sigma(n_d), n_d^T)$ where sim is a similarity score based on features representative of the functionality of the neuron. We explore three options for computing the similarity score: a) the (negative) Euclidean distance between the weights of the neurons, b) the (negative) Euclidean distance between the output activations of the neurons, and c) the Pearson correlation coefficient between the output activations of the neurons, computed from 500 records [199]. We use the Hungarian algorithm [187] to find the optimal permutation.

The final transformation is a sequence of permutations σ_l, one for each layer $l = 1, \ldots, L$. The permutations can be applied in a bottom-up or top-down order.

Bottom-up order. For each layer l ranging from 1 to $L - 1$, we perform the following steps:

(S1) Compute the optimal assignment σ_l between neurons in the l-th layer of $\mathcal{M}$ and $\mathcal{M}^T$. Note that under option a), we use as features the *input weights*, as the *output weights* point towards neurons in layer $l+1$ that are not yet re-aligned between $\mathcal{M}$ and $\mathcal{M}_T$.

(S2) In the current layer l, we permute the rows of the weight matrix $W^l(\mathcal{M})$ and bias vector $b^l(\mathcal{M})$ according to σ_l.

(S3) In the next (upper) layer $l+1$, we permute the columns of the weight matrix $W^{l+1}(\mathcal{M})$ according to σ_l.

Top-down order. For each layer l ranging from L down to 2, we perform the following steps:

(T1) Same as (S1), except that under option a), we use as features the *output weights* of the neuron, as the *input weights* are not yet re-aligned between $\mathcal{M}$ and $\mathcal{M}_T$.

(T2) In the current layer l, we permute the columns of the weight matrix $W^l(\mathcal{M})$.

(T3) In the next (lower) layer $l-1$, we permute the rows of the weight matrix $W^{l-1}(\mathcal{M})$ and bias vector $b^{l-1}(\mathcal{M})$.

Note that bottom-up weight-based re-alignment leads to different internal representations of $\mathcal{M}$ compared to top-down weight-based re-alignment, while bottom-up activation-based (correlation-based) re-alignment leads to the same result as top-down activation-based (correlation-based) re-alignment. Indeed, the similarity scores under b) and c) only depend on the output activations of the l-th layer, which are not impacted by the permutation applied to the previous layer (i.e., $l-1$ for bottom-up order and $l+1$ for the top-down order). However, the layer order impacts the final result under a), as the ordering of weights of a neuron is impacted by the permutation applied to the previous layer.

These algorithms can be easily extended to convolutional layers, replacing neurons by filters, activations by activation maps, etc. However, special care is required when propagating a permutation to the next layer if the two layers are of different types, i.e., at the junction between convolutional and FC layers in CNNs. To see how this impacts the algorithms, take

for instance bottom-up re-alignment. If the current layer (to be permuted) l is convolutional while the upper layer $l+1$ is fully connected, steps (S1) and (S2) would be the same, except that they would be applied to filters instead of neurons. However, we replace step (S3) as follows. First, we group the columns of the weight matrix $W^{l+1}(\mathcal{M})$ by the input filter, reversing the flattening operation applied to the activation maps before they are fed to the FC layer. Second, we permute the groups of columns according to σ_l. Finally, we flatten the groups back to a list of columns, thus restoring the 2-D shape of the matrix.

Note that *perfect re-alignment is not guaranteed*, due to (1) the heuristic metrics used and (2) potential errors being propagated, and accumulated, from the first to last layer re-aligned.

Re-alignment after initialisation. Guided by the insight that model initialisation is the main factor responsible for shadow model misalignment, we experiment with shadow model re-alignment after their initialisation. More specifically, each shadow model is (1) initialised, (2) re-aligned to the target model $\mathcal{M}_T$ using top-down weight-based re-alignment, and (3) trained. The only difference with the standard training of shadow models is the addition of step (2). We used top-down weight-based re-alignment due to its better overall performance (see Sec. 7.8.3).

Implementation details. The re-alignment techniques based on activations are performed using 500 records, which we select to be different from the R records used to compute the misalignment scores (see Sec. 7.6.3 for details).

Baseline. We further implement the weight sorting technique of Ganju et al. [131]. This approach maps a model to its *canonical representation*, by sorting its layers in a bottom-up order, according to the sum of input weights of neurons. We extend this approach to convolutional layers, and map the shadow models as well as the target model to their canonical representation.

7.7.2 Re-alignment results

Table. 7.4 shows that one-to-one matching techniques (A3)-(A6) can effectively reduce the misalignment consistently across the three metrics considered. As expected, the method performing

Table 7.4: **Misalignment scores for different re-alignment techniques** applied to shadow models trained by the classical adversary.

Re-alignment technique	First layer (20 filters)	Second layer (50 filters)	Third layer (500 neurons)	Fourth layer (10 neurons)
A. Weight misalignment scores (↓ indicates better alignment)				
(A0) No re-alignment	12.44 (0.36)	16.56 (0.20)	30.98 (0.50)	12.87 (0.28)
(A1) Weight sorting [131]	11.51 (0.48)	16.28 (0.29)	30.96 (0.50)	12.74 (0.24)
(A2) Re-alignment after initialisation	12.49 (0.16)	16.00 (0.14)	30.39 (0.53)	8.56 (0.21)
(A3) Bottom-up weight-based re-alignment	**8.87** (0.56)	**14.19** (0.19)	**29.35** (0.44)	11.01 (0.21)
(A4) Bottom-up activation-based re-alignment	9.45 (1.00)	14.51 (0.07)	30.13 (0.45)	10.54 (0.01)
(A5) Top-down weight-based re-alignment	10.80 (0.81)	14.98 (0.24)	30.37 (0.48)	**5.71** (0.04)
(A6) Bottom-up correlation-based re-alignment [199]	9.38 (0.95)	14.58 (0.11)	30.45 (0.49)	11.49 (0.06)
B. Activation misalignment scores (↓ indicates better alignment)				
(A0) No re-alignment	68.64 (1.93)	72.13 (0.29)	27.90 (0.45)	0.42 (0.03)
(A1) Weight sorting [131]	60.64 (0.73)	70.19 (0.58)	27.81 (0.36)	0.42 (0.03)
(A2) Re-alignment after initialisation	69.92 (1.64)	70.66 (0.26)	26.95 (0.25)	0.42 (0.02)
(A3) Bottom-up weight-based re-alignment	43.22 (4.73)	59.10 (0.82)	25.79 (0.22)	0.42 (0.03)
(A4) Bottom-up activation-based re-alignment	**40.57** (3.01)	55.62 (0.50)	**24.07** (0.27)	0.42 (0.03)
(A5) Top-down weight-based re-alignment	48.07 (4.26)	62.30 (0.81)	26.26 (0.34)	0.42 (0.03)
(A6) Bottom-up correlation-based re-alignment [199]	**40.57 (3.02)**	**55.63 (0.57)**	24.36 (0.28)	0.42 (0.03)
C. Activation correlation scores (↑ indicates better re-alignment)				
(A0) No re-alignment	0.14 (0.03)	0.02 (0.01)	0.01 (0.00)	0.28 (0.08)
(A1) Weight sorting [131]	0.29 (0.02)	0.07 (0.01)	0.01 (0.00)	0.28 (0.08)
(A2) Re-alignment after initialisation	0.08 (0.01)	0.05 (0.01)	0.06 (0.01)	0.28 (0.08)
(A3) Bottom-up weight-based re-alignment	0.52 (0.07)	0.31 (0.02)	0.12 (0.01)	0.28 (0.08)
(A4) Bottom-up activation-based re-alignment	0.56 (0.05)	**0.39** (0.02)	**0.20** (0.01)	0.28 (0.08)
(A5) Top-down weight-based re-alignment	0.45 (0.07)	0.25 (0.01)	0.08 (0.00)	0.28 (0.08)
(A6) Bottom-up correlation-based re-alignment [199]	**0.57** (0.05)	0.29 (0.02)	**0.20** (0.01)	0.28 (0.08)

best in a given metric is the one optimizing over features used by the metric. Similarly, top-down re-alignment performs best in the upper layers, perhaps due to cumulative error effects as the algorithm progresses to the lower layers (with similar trends for bottom-up re-alignment). Top-down weight-based matching (A5) achieves the most spectacular improvement relatively to the no re-alignment baseline, reducing the weight misalignment score from 12.87 to 5.71. Bottom-up correlation matching (A6) is effective in reducing the misalignment, particularly in the lower layers.

We observe an interesting phenomenon in the third layer. The weight misalignment score (WMS) is only slightly reduced, from 30.98 to 29.35 using bottom-up weight matching (Table 7.4A). Yet this layer does become more aligned, since activation-based re-alignment metrics improve significantly: their misalignment score decreases from 27.90 to 24.07 (Table 7.4B), while their correlation score improves from 0.01 to 0.20 (Table 7.4C). We attribute the small reduction in WMS to the third layer being large (625500 weights) and placed in the middle of

the network, making it sensitive to alignment errors propagated from the other layers.

The weight sorting approach (A1) of Ganju et al. [131] performs poorly, on par with and sometimes slightly better than the no re-alignment baseline. We attribute its poor performance to the neuron sorting criterion: the sum of weights may be too coarse to allow for effective re-alignment. More generally, canonical representations have been argued to be sensitive to small changes in weights [21].

Finally, the re-alignment after initialisation technique (A2) proves to be ineffective, likely due to the inherent noise of matching the yet to be trained shadow model with the fully trained target model.

Summary. The consistent improvements of re-alignment techniques (A3)-(A6) confirm that it is possible to reduce the misalignment between the target model and shadow models.

7.8 White-box membership inference

A final important criterion to judge the effectiveness of re-alignment techniques is their performance on white-box attacks. In this section, we focus on white-box membership inference as a case study and perform a comprehensive and systematic evaluation of attacks. More specifically, for a complete understanding of the impact of re-alignment, we consider three threat models:

(S1) Auditor setup. Here, the adversary is assumed to have access to a large fraction f of the private dataset D_T (members), together with a dataset of non-members, i.e., records from the same underlying distribution $\mathcal{D}$ that do not belong to D_T. Introduced by Nasr et al. [229], the auditor extracts white-box features of member and non-member records *directly from the target model*. There is no need to train shadow models in this setup, as the signal available through the target model is, by definition, perfectly aligned. We include this setup because it acts as an upper bound for the accuracy of white-box shadow modelling attacks based on a particular set of features. Note that the attack performance can still vary depending on the

meta-classifier architecture used and fraction f of data available.

(S2) Same model initialisation adversary. We consider an adversary with the ability to train shadow models having the same initialisation as the target model $\mathcal{M}_T$. We additionally assume the adversary to have access to a shadow dataset D_{shadow} from the same distribution $\mathcal{D}$ that is disjoint from D_T. This adversary is informed by the findings of Sec. 7.6 that model initialisation (MI) is the main reason for shadow model misalignment and that shadow models with same MI are more aligned to $\mathcal{M}_T$.

(S3) Classical adversary. This is the same as (S2), except that the adversary does not know the initialisation of the target model $\mathcal{M}_T$ (see Sec. 7.2 for further details). We further evaluate the performance of the classical adversary using the re-alignment techniques presented previously (S4)-(S9).

7.8.1 Evaluation setup

In each setup, we train a machine learning meta-classifier to perform the membership inference task. The meta-classifier takes as input a set of features $F(\mathcal{M}, x)$ computed by feeding a record x through a model $\mathcal{M}$, together with the membership label of the record. We consider three types of features: (1) output activations (OA) x^l computed at a given layer, (2) gradients (G) of the loss function computed on the record (x, y) with respect to the weights of a layer l: $\frac{\partial \mathcal{L}}{\partial W^l_{i,j}}(x, y), i = 1, \ldots, D^l, j = 1, \ldots, D^{l-1}$, and (3) input activations $(W^L_{y,i} x^{L-1}_i)_{i=1,\ldots,D^{L-1}}$ entering the neuron that encodes the ground-truth class y. Features in (3) can be seen as a finer grained version of the signal encoded by the model confidence for class y. (1) and (2) were first proposed by Nasr et al. [229], while (3) have been used by Leino et al. [193] to learn a displacement function with respect to the same features computed on a proxy model. Our use of (3) is different and simpler, as we combine these features directly with (1) and (2) and show them to enhance MIAs.

Given a dataset D, we partition it randomly into two datasets of equal size, D_{shadow} and D_{test}. The private, target dataset D_T is obtained by sampling $|D_{\text{test}}|/2$ records from D_{test}, uniformly

at random without replacement. We randomly partition D_{test} into a meta-classifier test set of size N_{test} on which we evaluate all attacks and setups, and two meta-classifier train and validation sets, $D_{\text{train}}^{\text{auditor}}$ and $D_{\text{val}}^{\text{auditor}}$, made available to the auditor in setup (S1). As the partitioning is random, all the datasets consist of half members and half non-members of D_T. Let $N_{\text{test}}, N_{\text{train}}^{\text{auditor}}$, and $N_{\text{val}}^{\text{auditor}}$, denote their respective dataset sizes. In setups (S2)-(S9), the adversary has access to D_{shadow} (by design, disjoint from D_{test}), which we randomly partition into $N_{\text{train}}^{\text{shadow}}$ and $N_{\text{val}}^{\text{shadow}}$ records used to respectively train and validate the meta-classifier.

In setup (S1), we train the meta-classifier on features computed directly on the target model $\mathcal{M}_T$ and $N_{\text{train}}^{\text{auditor}}$ records. In setups (S2)-(S9), we train K shadow models $\mathcal{M}_k$, for $k = 1, \ldots, K$, each on a different dataset D_k of size $|D_T|$ sampled uniformly at random without replacement from D_{shadow}.

Meta-classifier details. We use a meta-classifier architecture similar to the one proposed by Nasr et al. [229]. It consists of multiple modules, each used to separately embed the different types of features sets in a layer, and a label embedding model. The outputs of these modules are concatenated together and passed through a membership classifier module. When the features used are gradients, we embed the gradients separately in each layer using a CNN architecture, consisting of a convolutional layer with kernel size equal to 100. The CNN is followed by a dropout layer with probability 0.2 and an MLP with a hidden layer of size 128, ReLU non-linearity, and an output layer of size 64. When the features used are activations, we embed the activations separately in each layer using the same MLP architecture as described above. We embed each of the class label of the record using a learnable weight vector of size 16. Finally, we use an MLP architecture as the membership classifier. We use two hidden layers of sizes 128 and 64, each followed by ReLU non-linearity, and an output layer of size 2.

The meta-classifier is trained end-to-end using mini-batch gradient descent on the binary cross-entropy loss. We use the Adam optimiser [180], a learning rate of $\eta' = 0.001$, and train the model for a maximum of 100 epochs. We divide η' by 2 at the end of an epoch if the validation accuracy does not improve compared to the best found so far, and stop the training when $\eta' < 0.0001$. The validation records are passed through the target model $\mathcal{M}_T$ in setup (S1)

and through a *validation shadow model* in setups (S2)-(S9). In other words, the meta-classifier is trained on feature vectors derived from $K-1$ shadow models and early stopped based on the performance on the K-th shadow model. For better performance, each mini-batch only contains features computed from a single shadow model. Each epoch consists of one pass over all the unique records in the training dataset, where we randomise the selection of the shadow model used in a batch. The validation shadow model is, by default, the first among the K trained for all the datasets with one exception. On the Texas100 dataset, we observe a high variability between shadow models in terms of the number of completed training epochs N_{best} before the best model is found. Among the K shadow models, we select the one having the median value of N_{best} as the validation shadow model.

7.8.2 Datasets and models

We evaluate the attacks on three real-world datasets commonly used to evaluate privacy risks [302, 283, 229].

CIFAR10 [186]. From the 60000 records available, we set aside two disjoint datasets of size $V = 2000$ records each to be used as validation data for shadow and target models, respectively. This leaves us with a dataset D of 56000 records which we process as described in Sec. 7.8.1. We use the same CNN architecture as described in Sec. 7.6.3.

Texas100. This dataset[1] consists of 67330 records and 6169 attributes. Each record is labeled with one of 100 classes. We randomly select 44000 records, set aside $2V$ validation records as described above, leaving us with a dataset D of 40000 records. We use the 5-layer MLP from Nasr et al. [229], having hidden layer sizes of 1024, 512, 256, and 128 and an output layer of size 100. The hidden layers are followed by ReLU non-linearity and a dropout layer having $p = 5\%$.

Purchase100. This dataset consists of 197324 records and 600 attributes. Each record is labeled with one of 100 classes. We randomly select 84000 records, set aside $2V$ validation

[1]We use the pre-processed Texas100 and Purchase100 datasets made available at this link: `https://github.com/privacytrustlab/datasets`

records as described above, leaving us with a dataset D of 80000 records. We use a 4-layer MLP having layer sizes of 512, 256, 128 and an output layer of size 100. The hidden layers are followed by ReLU non-linearity.

Across datasets and models, the training details are the same as the ones used in Sec. 7.6.3. Across experiments, we set the number of meta model test, validation, and train records to be $N_{\text{test}} = 5000$, and $N_{\text{val}}^{\text{auditor}} = N_{\text{val}}^{\text{shadow}} = 2000$, and $N_{\text{train}}^{\text{auditor}} = N_{\text{train}}^{\text{shadow}} = |D|/2 - N^{\text{test}} - N_{\text{val}}^{\text{auditor}}$, respectively. We use $K = 10$ shadow models, repeating the experiment on $R = 10$ different target models trained with different initialisations. We report the Area Under the Curve (AUC) of the meta-classifier computed on the N_{test} records (mean and 95% confidence interval over $R = 10$ runs).

7.8.3 Results

We first analyse the impact of re-alignment on individual, internal layers of the CNN model.

Table. 7.5 illustrates the white-box paradox on the output activations (OA) of the second to last layer. As we move from the powerful auditor (S1) to the weaker adversaries, the AUC drops from 0.805 (S1) to 0.750 when the adversary knows the target model initialisation (S2). Lacking this assumption, the classical adversary training the meta-classifier on features derived from misaligned shadow models achieves random performance, at 0.509 (S3). However, the top-down weight matching technique is able to fully recover the signal, reaching 0.813 (S7). Except weight sorting [131], all other methods improve the AUC of MIAs.

The gradients are also affected by misalignment, although the drop in AUC when moving from (S1) to (S3) is smaller than for OA. This is likely due to gradients of members and non-members being easier to distinguish due to differences in norm [229], something which the general meta-classifier can learn in spite of misalignment. Even so, re-alignment techniques are again able to recover the signal, e.g., top-down weight matching reaches 0.834 (S7), much better than 0.789 (S3) using no re-alignment, and comparable to the auditor, at 0.837 (S1). As expected, the MIA performance of the same model initialisation adversary (S2) is less affected by the

Table 7.5: **CIFAR10: Results of white-box MIA using features from individual layers.** We report the mean AUC with 95% confidence interval. We perform attacks using output activations (OA) or gradients (G) of different layers. Our re-alignment techniques (S5)-(S9) successfully recover the membership signal in each of the individual layers, outperforming the naïve classical adversary (S3).

Threat model \ Layer	Last	Second to last		Third to last	Fourth to last
	G	OA	G	G	G
(S1) (Auditor) Partial access to target dataset	0.837 ± 0.012	0.805 ± 0.014	0.810 ± 0.017	0.820 ± 0.016	0.834 ± 0.010
(S2) Same model initialisation adversary	0.822 ± 0.012	0.750 ± 0.012	0.823 ± 0.015	0.811 ± 0.015	0.830 ± 0.009
(S3) Classical adversary	0.789 ± 0.021	0.509 ± 0.005	0.806 ± 0.014	0.788 ± 0.019	0.768 ± 0.024
(S4) Classical adversary + weight sorting [131]	0.789 ± 0.025	0.511 ± 0.006	0.803 ± 0.013	0.791 ± 0.020	0.770 ± 0.023
(S5) Classical adversary + bottom-up weight-based re-al.	0.795 ± 0.017	0.581 ± 0.029	0.818 ± 0.017	0.802 ± 0.019	0.789 ± 0.022
(S6) Classical adversary + bottom-up act.-based re-al.	0.797 ± 0.023	0.652 ± 0.007	0.820 ± 0.015	**0.803** ± 0.018	**0.793** ± 0.024
(S7) Classical adversary + top-down weight-based re-al.	**0.834** ± 0.013	**0.813** ± 0.010	**0.822** ± 0.019	0.800 ± 0.020	0.787 ± 0.021
(S8) Classical adversary + bottom-up corr.-based re-al.	0.795 ± 0.013	0.639 ± 0.014	0.818 ± 0.017	0.803 ± 0.019	0.796 ± 0.021
(S9) Classical adversary + re-aligned after initialisation	0.818 ± 0.014	0.757 ± 0.015	0.809 ± 0.018	0.786 ± 0.019	0.768 ± 0.023

Table 7.6: **Our input activation (IA) features improve the performance of white-box MIAs.** We report results in the (S1) Target dataset (auditor) scenario.

Dataset	White-box			Black-box
	OA + IA + G (Ours)	OA + G (Nasr et al. [229])	OA + IA (Ablation)	OA (Shokri et al. [302])
CIFAR10	**0.860** ± 0.013	0.839 ± 0.012	0.846 ± 0.010	0.816 ± 0.010
Texas100	**0.912** ± 0.020	0.904 ± 0.021	0.907 ± 0.019	0.880 ± 0.015
Purchase100	**0.757** ± 0.013	0.739 ± 0.012	0.739 ± 0.014	0.733 ± 0.010

misalignment, reaching 0.822. We observe similar trends in the third and fourth to last layers, although none of the re-alignment techniques fully recovers the signal. This is however not an issue in MIAs, as the useful signal generally lies in the top layers [229].

We show in Table 7.6 that input activations (IA) extracted from the last layer improve the performance of white-box MIAs consistently across datasets. To minimise the effect of misalignment and isolate the improvement due to IAs alone, we perform a detailed ablation in the auditor setup (S1). Our results suggest that IAs add new membership signal that is not available in the gradients. On CIFAR10 for instance, the AUC increases from 0.816 (black-box, OA) to 0.860 (white-box, OA + IA + G), better than OA + G (white-box [229]), at 0.839. In light of these results, we believe that IAs, which are simple and inexpensive to compute, should be incorporated in future ML privacy risks assessments.

Next, we show results of white-box MIAs that combine features from more layers, progressively adding new layers from top to bottom. This includes as a particular case the black-box attack that uses the output activations (OA) [302]. The white-box attacks combine the OAs and

Table 7.7: **CIFAR10: Results of white-box MIAs using different threat models and re-alignment methods.**

Threat model	Black-box	White-box		
	Last layer	Last layer	Last two layers	All four layers
(S1) (Auditor) Partial access to target dataset	0.816 ± 0.010	0.860 ± 0.013	0.857 ± 0.014	0.853 ± 0.015
(S2) Same model initialisation adversary	0.810 ± 0.011	0.857 ± 0.013	0.859 ± 0.019	0.862 ± 0.018
(S3) Classical adversary	0.815 ± 0.011	0.837 ± 0.014	0.837 ± 0.015	0.842 ± 0.015
(S4) Classical adversary + weight sorting [131]	0.813 ± 0.012	0.838 ± 0.014	0.840 ± 0.015	0.844 ± 0.015
(S5) Classical adversary + bottom-up weight-based re-al.	0.813 ± 0.012	0.853 ± 0.014	0.850 ± 0.018	0.855 ± 0.018
(S6) Classical adversary + bottom-up act.-based re-al.	0.815 ± 0.012	**0.860** ± 0.014	0.861 ± 0.017	0.860 ± 0.017
(S7) Classical adversary + top-down weight-based re-al.	0.815 ± 0.013	0.849 ± 0.017	0.851 ± 0.017	0.856 ± 0.019
(S8) Classical adversary + bottom-up corr.-based re-al.	0.812 ± 0.010	0.858 ± 0.015	**0.861** ± 0.015	**0.863** ± 0.018
(S9) Classical adversary + re-aligned after initialisation	0.811 ± 0.010	0.844 ± 0.015	0.843 ± 0.019	0.844 ± 0.016

gradients (G) of each layer with the input activations (IA) of the last layer. Table 7.7 shows that the performance of white-box attacks is affected by shadow model misalignment, as it drops from 0.860 in the strong, auditor scenario (S1) to 0.837 in classical adversary scenario (S3). In contrast, the black-box attack is not affected by shadow model misalignment, as its accuracy is roughly constant at 0.815. The re-alignment techniques, and in particular the bottom-up correlation matching, are able to recover the lost signal, increasing the performance to e.g., 0.858 using features in the last layers and 0.863 using features in all the layers. The performance of the same model initialisation adversary (S2) is similar to the auditor (S1), suggesting that **releasing the model initialisation is likely to increase the risk of attacks**.

Table 7.8 shows that when applied to an MLP trained on Texas100, the classical and black-box adversaries perform similarly, reaching 0.857 and 0.859, respectively. Re-aligning the shadow models however increases the performance of the classical adversary to 0.878 (bottom-up weight matching) (S5), with similar results for bottom-up activation matching (S6) and top-down weight matching (S7) (0.871 and 0.876, respectively). We observe a noticeable performance gap compared to the auditor (S1) (0.912). We also noticed that the training procedure stopped much earlier for some shadow and target models compared to others. We therefore suspect the gap might be due to higher variability between models which might increase misalignment.

Table 7.9 reports attack results for MLP trained on Purchase100. The gap between the auditor (S1) and the classical adversary (S3) is smaller than in previous datasets (0.757 vs 0.745), and the re-alignment techniques do not significantly improve the attack performance. We attribute

Table 7.8: **Texas100: Results of white-box MIAs using different threat models and re-alignment methods.** We show results of attacks using features from the last layer. For brevity, we omit the results of the black-box attack for (S3)-(S8) as they are identical to (S3), due to re-alignment techniques not impacting the output layer.

Threat model	**Black-box (OA)**	**White-box (OA + IA + G)**
(S1) (Auditor) Partial access to target dataset	0.880 ± 0.015	0.912 ± 0.020
(S2) Same model initialisation adversary	0.843 ± 0.023	0.873 ± 0.041
(S3) Classical adversary	0.859 ± 0.015	0.857 ± 0.048
(S4) Classical adversary + weight sorting [131]	-	0.866 ± 0.041
(S5) Classical adversary + bottom-up weight-based re-al.	-	**0.878** ± 0.029
(S6) Classical adversary + bottom-up activation-based re-al.	-	0.871 ± 0.045
(S7) Classical adversary + top-down weight-based re-al.	-	0.876 ± 0.039
(S8) Classical adversary + bottom-up correlation-based re-al.	-	0.859 ± 0.050

Table 7.9: **Purchase100: Results of MIAs against an MLP using different threat models and re-alignment methods.**

Threat model	**Black-box (OA)**	**White-box (OA + IA + G)**
(S1) (Auditor) Partial access to target dataset	0.733 ± 0.010	0.757 ± 0.013
(S2) Same model initialisation attacker	0.736 ± 0.010	0.745 ± 0.012
(S3) Classical attacker	0.733 ± 0.012	0.746 ± 0.011
(S4) Classical attacker + weight sorting [131]	-	0.745 ± 0.012
(S5) Classical attacker + bottom-up weight-based re-al.	-	0.750 ± 0.013
(S6) Classical attacker + bottom-up act.-based re-al.	-	0.746 ± 0.014
(S7) Classical attacker + top-down weight-based re-al.	-	0.749 ± 0.013
(S8) Classical attacker + bottom-up corr.-based re-al.	-	0.744 ± 0.013

these findings to the model being smaller, regularised with dropout, and thus more robust overall.

Worst-case privacy. Carlini et al. [52] have recently argued that attacks should be considered successful whenever they are able to confidently target at least a few records, even if their accuracy is low. To understand our attack's performance in high confidence settings, we report the complete Receiver Operator Characteristic (ROC) curves of attacks on a log-log scale, comparing the various alignment techniques according to the True Positive Rate (TPR) for low values of the False Positive Rate (FPR). The smallest FPR we consider is 1% as using a smaller value, e.g., 0.1% would lead to a noisy value of TPR. We average the ROC curves over 10 experiments. Fig. 7.6a shows that on the CIFAR10 dataset and in terms of the TPR for an FPR of 1%, our white-box approach using re-alignment achieves 12.3% and performs better than (1) the black-box attack of Shokri et al. [302] (7.2%), (2) the white-box attacks with no re-alignment (7.8%), and (3) the weight sorting technique of Ganju et al. [131] (7.8%). Fig. 7.7b

and c report similar results on Texas100 and Purchase100 datasets, although the differences are less marked.

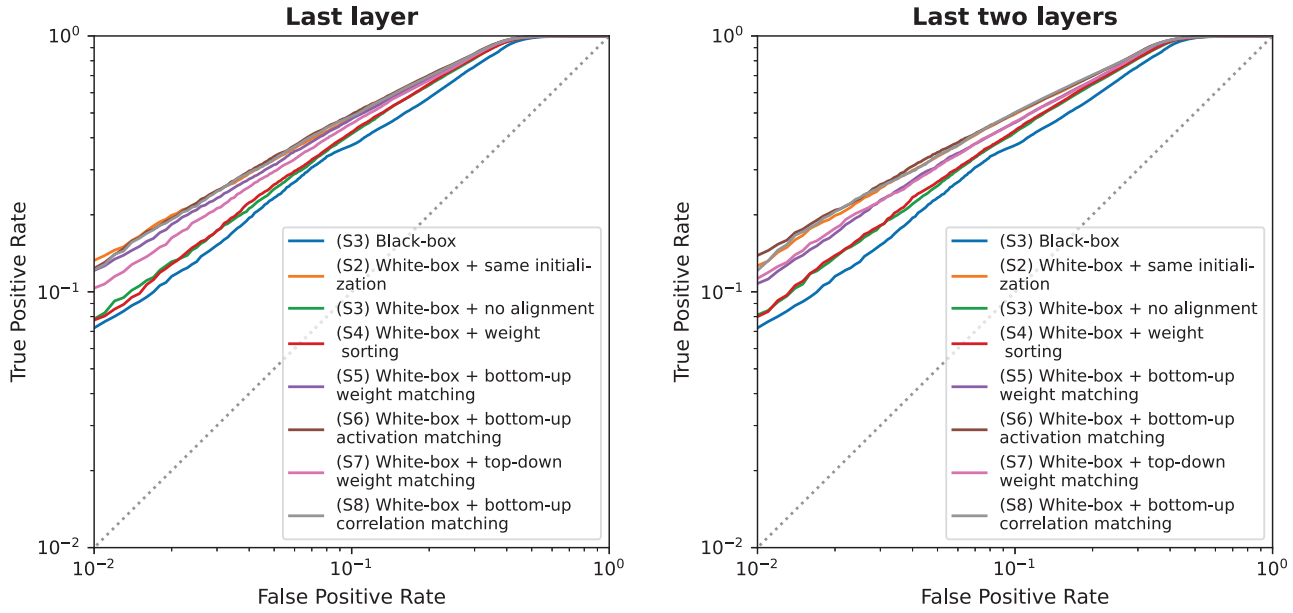
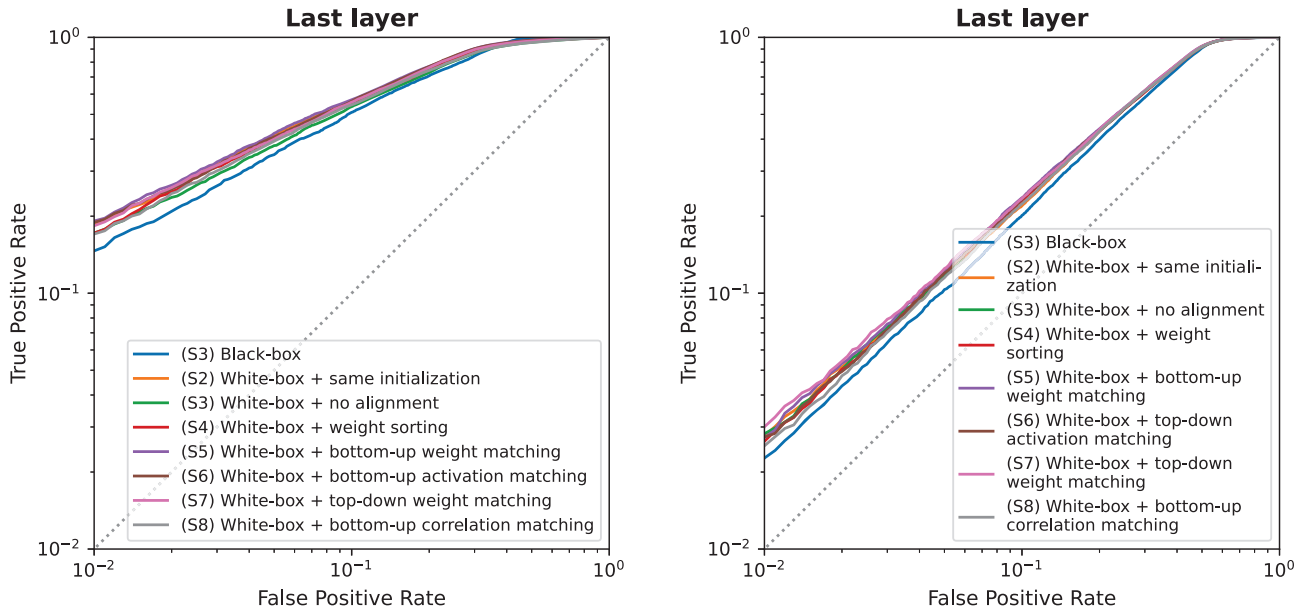


Figure 7.6: **ROC curves of attacks on the CIFAR10 dataset.** The white-box attacks use features extracted from the last layer (left) or from the last two layers (right). We also report results for the black-box attack using the output activations of the last layer.



(a) Texas100 dataset

(b) Purchase100 dataset

Figure 7.7: **ROC curves of attacks on other datasets.**

Comparison with set-based classifiers. An alternative approach to deal with the permutation equivalence symmetry, which was explored in prior works on property inference attacks

(PIA) [131, 318], is the use of set-based classifiers. A set-based classifier views the neurons in a layer as a set and aggregates their feature representations using a permutation-invariant operation such as summation. In PIAs, a set-based classifier is trained on the *weights* θ of a model f_θ to infer a property of the training dataset, e.g., the proportion of females in the dataset. We here extend set-based classifiers to white-box MIAs, using activations and gradients as features instead of the weights. More specifically, we restrict to neurons in the last layer L, which contain most of the membership signal [229]. For $d = 1, \ldots, D^{L-1}$, we represent how the dth neuron acts on a record (x, y) using the following feature vector $v_d(x) \in \mathbb{R}^{D^L+3}$: (1) the output activation x_d^{L-1}, (2) the input activation entering the neuron of the correct label $W_{y,d}^L x_d^{L-1}$, (3) the gradients of the weights coming out the neuron $\partial\mathcal{L}/\partial W_{i,d}^L(x, y), i = 1, \ldots, D^L$, and (4) the gradient of the bias term $\partial\mathcal{L}/\partial b_d^{L-1}(x, y)$. We embed $v_d(x)$ using an MLP ϕ with one hidden layer of size 128 and an output size of 64. We train a meta-classifier (MC) on the concatenation of the set representation $\sum_{d=1}^{D^{L-1}} \phi(v_d(x))$, the embedding of the output activations of the last layer x^L, and the embedding of the label y, with all other details the same as before.

Table 7.10 shows that on the CIFAR10 dataset, the set-based classifier performs on par with the classical adversary (S3) without re-alignment and less well than our re-alignment techniques. On Texas100, the set-based classifier performs similarly to our re-alignment techniques, recovering part of the MIA signal, but less well than the idealised auditor setting (S1). We believe these results can be explained by the difficulty of training set-based classifiers for large set sizes (i.e., number of neurons), as the signal coming from individual neurons is more "smoothed", i.e., averaged out. Indeed, the CIFAR10 dataset has 500 neurons in the second to last layer, while Texas100 has only 128 neurons, making it harder to effectively train a set-based classifier in the former case.

Comparison with prior work. We compare our approach with the Bayes-WB approach of Leino et al. [193]. This approach trains proxy models whose goal is to replicate the "semantic meaning" of the target model features in an internal layer l. To this end, the proxy model is initialised using the target model weights, then all the layers up to l are frozen, while the remaining layers are re-trained from scratch on shadow data. Then, for each target record (x, y), they compute local linear approximations $(W_T^x, b_T^x), (W_1^x, B_1^x)$ of how the remaining layers

Table 7.10: **Results of MIAs using a set-based classifier.**

CIFAR10	Texas100	Purchase100
0.840 ± 0.014	0.873 ± 0.025	0.747 ± 0.013

Table 7.11: **CIFAR10: Comparison with Leino et al. [193].**

Method	MIA AUC
Bayes-WB (Leino et al. [193])	0.708 ± 0.005
IA only, not re-aligned (S3)	0.726 ± 0.006
IA only, re-aligned (S7)	**0.805** ± 0.014

restricted to the output class y act on the input x, in the target and proxy model, respectively. Finally, they use a Bayesian approach exploiting the differences between the behaviour of the linear approximations on the target record, to produce a membership score: $sigmoid((W_T^x - W_1^x)^t x + (b_T^x - b_1^x))$. This is shown to be optimal on binary logistic regression under specific assumptions on the data distribution $\mathcal{D}$.

We re-implement their approach using the last layer, which is linear, meaning that we can use the pure signal, without the need to compute local approximations. Indeed, we were unable to replicate the influence-based linear approximators. In this scenario, the Bayes-WB approach uses as features the IAs. We train $K = 10$ proxy models and use their average weights as recommended by the authors. For a fair comparison, we also run our re-alignment-based attack using the same features. To minimise the re-alignment error in the top layer, we choose top-down weight matching (S7).

Table 7.11 shows that Bayes-WB performs worse than the classical adversary using the input activations (IA), reaching 0.708 and 0.726, respectively on the CIFAR10 dataset. Our re-alignment technique achieves much better performance (0.805). We conclude that although proxy models are designed to be "aligned" in some sense with the target model at the last layer, they do not replicate well the dataset memorisation patterns of the target model, likely due to only the last layer being trained on the dataset while the lower layers are maintained frozen.

7.9 Discussion

In this section, we discuss in more detail our assumptions and outline potential avenues for extending our work.

Black-box vs white-box MIA gap. Although we show re-alignment to improve white-box MIAs, our results suggest that their advantage remains in general and on average small compared to black-box. Our results on the strongest adversaries (the auditor and the same model initialisation adversaries) however show that this is not a limitation of re-alignment techniques. Indeed, even the strongest adversary, which does not suffer from the misalignment issue, only performs, e.g., 5 p.p. better on CIFAR10. Our intuition is that a single record does not have much impact on the weights of the model and thus the membership signal mostly lies in the output layer. Interestingly, this is different from other inference settings where the gap between white-box and black-box attackers has been found to be much larger. For instance, we report in Chapter 6 a correlation inference accuracy gap of more than 25% between white-box and black-box attackers knowing the model initialisation seed (Fig. 6.11). This strongly suggests that model weights memorise information about the dataset distribution (in Chapter 6, correlation coefficients) that cannot be recovered from the outputs alone. More work is however needed to characterise the determinants of large white-box vs black-box performance gaps, i.e., what information from the private dataset is memorised in the internal layers of a model *and* cannot be recovered from the outputs alone.

We further note that the true advantage of the white-box adversary may be higher in real deployments. Indeed, realistic black-box adversaries are more restricted in their capabilities, as they may not have precise knowledge of the model architecture and the owner may limit the number of queries or the size of the prediction vector. Although the adversary could compensate for this coarser signal by asking more queries, each query incurs a cost in settings such as machine learning as a service.

Addressing other symmetries. Our work focuses on permutation equivalence, using standard alignment techniques from the ML literature. Future work could develop more general

alignment techniques dealing with all known symmetries. As we observe from the distance metrics of the re-aligned models, there is scope for designing better alignment techniques that will further reduce the distance between the shadow models and the target model.

Other architectures. We have focused on misalignment phenomena in MLPs and standard CNNs. A common feature of classification models, regardless of their architecture, is their use of feed-forward layers as the output classification module. Since the output layer is also where most of the membership signal lies [229], our re-alignment techniques apply beyond the architectures studied in this work. Beyond this, future work could analyse the symmetries present in other architectures, including more advanced CNNs, e.g., with skip connections [149] and Transformers [336].

Targeted white-box MIAs. The white-box MIAs considered in this work rely on a common meta-classifier for all the target records. Targeted attacks using a record-dependent decision boundary have been shown to perform even better in the black-box setting [353, 52]. We believe that targeted white-box attacks combined with our aligned shadow models are likely to further increase the risk. A potential and yet untapped advantage is the use of weights as attack features. Weights would be available since hundreds of models would typically be trained with or without the target record [52]. Whether weights contain new membership signal that is not redundant with the activations and gradients is an open question.

7.10 Conclusion

We here systematically investigate the causes of misalignment in the context of shadow models, extend and evaluate the effectiveness of re-alignment methods proposed in the model fusion literature to the shadow modelling context, and perform a comprehensive evaluation of white-box membership inference attacks (MIA). Our results show that (a) changing the weight initialisation is the main cause of misalignment, (b) while re-alignment methods successfully reduce the measured misalignment between the target and shadow models, they struggle with re-aligning large and middle layers, (c) white-box MIA using re-aligned shadow models improves

the TPR by 4.5% on the CIFAR10 dataset with an FPR of 1%. Our results highlight that on-device deployment can increase the risk as newly available information can be exploited by an attacker.

Chapter 8

Detection avoidance attacks against perceptual hashing client-side scanning

The work presented in based on the following peer-reviewed conference article: Jain, S., Crețu, A. M.*, and de Montjoye, Y.-A. (2022). "Adversarial Detection Avoidance Attacks: Evaluating the robustness of perceptual hashing-based client-side scanning". In 31st USENIX Security Symposium (USENIX Security 22) (pp. 2317-2334).* `https://www.usenix.org/conference/usenixsecurity22/presentation/jain`

This chapter is based on the revised paper with results obtained after removing duplicates from the ImageNet dataset. The revised paper is available as an arXiv preprint at `https://arxiv.org/abs/2106.09820`.

Shubham Jain and I are joint first authors on the paper. We collaborated in the design of the black-box method, in the design of the experiments, in the writing of the code, and in the writing of the paper. We equally shared the workload in writing the code, running the black-box experiments, generating figures and writing the paper. I led the white-box attack design and experiments and the removal of duplicates from the ImageNet dataset, while Shubham Jain led the diversity design and experiments.

8.1 Introduction

More than two billion people across the world use end-to-end encryption (E2EE)-enabled platforms such as Signal and WhatsApp [304, 37], exchanging more than 100 billion messages daily on WhatsApp alone [303]. E2EE provides a strong privacy protection to users, preventing governments, hackers, and platform providers themselves to access the content of their communications. Governments and organisations have however raised concerns that E2EE is preventing the detection of illegal content [231, 258] such as child sexual abuse media and terrorism-related content [244]. Governments have also recently been discussing bills looking at online safety including the countering of disinformation and misinformation online [120].

Following strong concerns from security experts and former national security officials alike [195, 122], key escrow systems providing governments an encryption "backdoor" have been abandoned. Instead, client-side scanning has been proposed to detect the sharing of illegal content on E2EE-enabled platforms by tech companies [17], researchers [145, 269, 211], and policy makers [76]. Here, a signature of a visual media (image, video) would be computed on the user's device and then compared against the database of signatures of known illegal images. If a match is found, the user would e.g. be flagged and the unencrypted content automatically shared for further review. Designed to be robust to small changes to the media, as well as transformations like rotation and rescaling, perceptual hashing algorithms would be used to generate the signature. Several variations of this scheme have been proposed. For instance, Apple's 2021 proposal [1] would have send a cryptographic voucher containing the encrypted images and the results of the match (the proposal has now been abandoned). If the number of successful matches reaches a predefined threshold, the system is then able to decrypt all the matched images.

A large literature has developed recently in adversarial machine learning, where the input of a machine learning classifier is adversarially modified to fool the model. Slightly modified but visually similar variants of an image can be designed resulting in a misclassification by the model [298, 203, 252]. Some of the most famous examples include an image of panda being misclassified as a gibbon [141] or a road works traffic sign to be misclassified as a give way

traffic sign [252]. A wide range of techniques in both white and black-box setup have been developed, including [209] and [301].

Contribution. We here propose the first framework to evaluate the robustness of perceptual hashing-based client side scanning to a novel *detection avoidance attack*. Specifically, our framework assesses the threat posed by adversarial attacks aiming to minimally modify an image so as to evade detection while preserving its content. Our framework also takes into account the diversity of modifications produced by the attack in terms of distances between signatures. The more diverse the perturbation, the harder it is for the system to mitigate by expanding the database with the modified images.

We propose three adversarial attacks against perceptual hashing algorithms, a general black-box attack, inspired by previous work in adversarial ML, e.g., [161, 284, 348], and two novel and optimal white-box attacks exploiting the linearity and orthogonality of discrete cosine transform (DCT)-based perceptual hashing algorithms. The attacks are designed to minimally perturb the image while avoiding detection. In particular, they produce a wide range of perturbations preventing easy mitigation strategies such as expanding the database with modified images.

We evaluate the robustness of perceptual hashing-based client-side scanning mechanisms and show them to be highly vulnerable to detection avoidance attacks. We perform a large-scale extensive evaluation of five commonly used perceptual hashing algorithms: pHash (continuous and discrete) [78], dHash, aHash and PDQ [178] and show that, for all of them, our attack manages avoid detection with modified images very similar to the original. Taken together, our results strongly suggest that perceptual hashing-based client-side scanning is highly vulnerable to small modifications in all scenarios considered. They also suggest that simple mitigation strategies like expanding the database would be ineffective. This sheds serious doubts on the robustness of currently proposed client-side scanning mechanisms based on perceptual hashing [145, 76, 269, 211].

8.2 Perceptual hashing-based client-side scanning

Perceptual hashing algorithms compute a signature of a visual media (e.g., image, video) without having to share it. Perceptual hashes are different from cryptographic hashes in that the former changes gradually as the image changes, while the latter changes significantly as soon as a single pixel changes. Importantly, perceptual hashes are designed to detect instances of a visual media that are visually similar (e.g., a resized version) without being exact copies [219]. To achieve this, they extract features that remain invariant under small modifications, such as resizing, noise addition, format change and rotation. For example, pHash and Facebook's PDQ use the discrete cosine transform (DCT) [12] to extract image features relating to lower frequencies in the DCT output, that remain invariant to image modifications such as blurring, resizing and watermarking.

Formally, a perceptual hashing algorithm $h : \mathcal{I} \to \mathcal{O}^l$ is a deterministic function mapping a media $X \in \mathcal{I}$ to a fixed-size vector representation, the *hash*, usually consisting of bits ($\mathcal{O} = \{0, 1\}$) or real numbers ($\mathcal{O} = \mathbb{R}$). The similarity between two media $X, X' \in \mathcal{I}$ is quantified by computing the *distance* between the hashes according to a metric, henceforth denoted by d. For bit-valued hashes, d is usually the Hamming distance, while for real-valued hashes a typical choice for d is the Euclidean distance.

Client-side scanning. Perceptual hashing-based client-side scanning (PH-CSS) for illegal image detection consists of a database $\mathcal{D} = \{X_1, \ldots, X_N\}$ of N images $X_i \in \mathcal{I}, \forall 1 \leq i \leq N$, a perceptual hashing algorithm h, a distance d, and a threshold $T > 0$. Given an image $X \in \mathcal{I}$, the detection system computes the distance between the hash $h(X)$ and the hashes of images in the database $h(X_i)$. The image is flagged if there exists $1 \leq i \leq N$ such that $d(h(X_i), h(X)) \leq T$. While in this work we study images, our framework is applicable more broadly to visual media (e.g., videos).

The detection system performance can be measured using the false positive and false negative rates [361].

False positive rate. For a threshold T and database $\mathcal{D}$, the *false positive rate* $FPR(T, \mathcal{D})$ is

defined as the probability that a media visually different from those in the database ($X \not\sim \mathcal{D}$) is detected:

$$FPR(T, \mathcal{D}) = P(\exists 1 \leq i \leq N \ : \ d(h(X), h(X_i)) \leq T | X \not\sim \mathcal{D}) \tag{8.1}$$

False negative rate. For a threshold T and database $\mathcal{D}$, the *false negative rate* $FNR(T, \mathcal{D})$ is defined as the probability that a media visually similar to medias in the database ($X \sim \mathcal{D}$) is wrongly rejected:

$$FNR(T, \mathcal{D}) = P(\forall 1 \leq i \leq N \ : \ d(h(X), h(X_i)) > T | X \sim \mathcal{D}) \tag{8.2}$$

For both the FPR and FNR, the randomness is taken over the distribution of media shared by users.

The threshold T controls the trade-off between the false positive and negative rates. Intuitively, as the threshold T decreases the system will wrongly flag fewer media outside the database. At the same time, it would wrongly reject more media visually similar to those in the database.

8.3 Attack model

8.3.1 Detection avoidance attack

We here propose an adversarial attack against perceptual hashing-based client-side scanning, which we call *detection avoidance attack*.

Attack model. We assume that a malicious agent, *the adversary*, is in possession of a image from the database, henceforth denoted *original image* $X \in \mathcal{D}$. The adversary's goal is to minimally modify X into X' such that its content is preserved while evading detection. More specifically, the adversary's goal is to modify X via an additive perturbation δ such that the modified image: (1) is valid, i.e., $X' = X + \delta \in \mathcal{I}$ meaning that no pixel goes out of bounds, (2) evades detection, i.e., $d(h(X), h(X + \delta)) > T$, where T is the threshold used by the detection

system and (3) the perturbation is minimal in terms of visual dissimilarity. We assume that the adversary knows the distance d and the threshold T. We revisit these assumptions later.

Assuming that the adversary quantifies the visual similarity between images using a metric v (with smaller values for higher visual similarity), the adversary's goal can alternatively be written as follows.

$$\text{Minimise:} \quad v(X, X+\delta) \tag{8.3}$$

$$\text{s.t.:} \quad d(h(X), h(X+\delta)) > T \tag{8.4}$$

$$X + \delta \in \mathcal{I} \tag{8.5}$$

The objective function quantifies the visual similarity between the original image X and the modified image $X' = X + \delta$. The adversary seeks to minimise this objective in order to preserve the image content. The first constraint requires that the image should evade detection by not being matched with the original image. However, it is possible the modified image might still be flagged because its hash is close ($\leq T$) to other images in the database. Finally, the second constraint requires that the modified image should be valid (within the image bounds).

Perturbation diversity requirement. Additionally, the attack should be resistant to simple defences that the system could implement, such as expanding the database with hashes of modified images. This assumption is realistic as the detection system might gain knowledge of the attack. The attack should therefore produce a wide range of random perturbations that cannot be predicted and added to the database.

Attack intuition. We provide three intuitions for why perceptual hashing-based client-side scanning could be vulnerable to the proposed detection avoidance attack.

First, by design, the perceptual hash of an image changes gradually as the image changes, opening up the possibility to find a minimally modified image X' such that $d(h(X), h(X')) > T$ while X and X' remain visually similar.

Second, mitigation strategies like increasing the threshold T or expanding the database with

hashes of modified images could lead to an increase in the detection system's false positive rate, rendering it unsuitable for the use case.

Third, the hash space likely contains many valid image hashes that are at least T away from the original image hash. Formally, the d-ball of radius T around the hash of the original image $B(X,T) = \{X' \in \mathcal{I} \ : \ d(h(X), h(X')) \leq T\}$ is such that its complement $\overline{B(X,T)}$ contains potentially many images X' for typical distances (Hamming, Euclidean) and suitable threshold values. Furthermore, common perceptual hashing algorithms are non-injective, meaning that multiple inputs can lead to the same output. This suggests that it may be possible to obtain several and possibly many different perturbations of the same image such that $d(h(X), h(X')) > T$.

Finally, PH-CSS is similar but different from a classification model. It is a threshold-based detection system for matches in a database. This poses unique challenges: first, as the threshold T can be quite large, it is not trivial to see why adversarial perturbations can be found that still preserve the original image content, and second, the detection system could expand the database with hashes of adversarial images produced using our attack, making perturbation diversity a core requirement.

8.3.2 Adversary access to the perceptual hashing algorithm

We consider two levels of adversary access to the perceptual hashing algorithm.

Black-box access. Unless otherwise specified, we assume that the adversary has black-box access to the perceptual hashing algorithm. In a black-box setting, the adversary can provide an input image X to retrieve its corresponding hash $h(X)$ but does not know how the algorithm works. We believe that this assumption is realistic in the context of perceptual hashing-based client-side scanning. Most of the proposed implementations of PH-CSS, including Apple's proposal [1], would compute the image hashes on the device [50, 17]. Our black-box assumption is further supported by the fact that Apple's model was made accessible in a recent iOS version, and by subsequent statements by Apple that this was "expected behaviour" [79].

We further assume that the adversary can retrieve output hashes for as many inputs as needed, without the hashes being uploaded to the server. This can be made possible for example by automating the image upload through the app and creating a honeypot to catch all the requests, read the hash sent in the request and send it back to the adversary.

White-box access. In some cases, the adversary could have full knowledge of the algorithm used by the detection system. This can for example be the result of reverse-engineering work and the limited number of available perceptual hashing algorithms available. This knowledge consists of the rules or transformations used by the algorithm to map the input to the output hash. In this work, we develop two novel attacks for DCT-based perceptual hashing algorithms such as the popular pHash and Facebook's PDQ (see Sec. 8.4.3). These attacks provide a strong attack targeting DCT as they are optimal in the sense of minimally modifying the input to evade detection. They also provide theoretical insights into the vulnerability of existing approaches.

8.4 Attack methodology

We present the methodology for a general black-box attack against perceptual hashing-based client side scanning and two novel attacks against DCT-based hashes.

8.4.1 Notation

General. We consider real-valued images $\mathcal{I} = [0,1]^n$ and define an image as an element $X \in I$. The image size n is equal to the number of pixels in the image. For $1 \leq i \leq n$, we use X^i to denote the i-th element of the (flattened) image X. Given an image $X \in \mathcal{I}$, the space of valid perturbations is denoted by $\mathcal{I} - X = \{\delta \in [-1,1]^n \ : \ 0 \leq \delta^i + X^i \leq 1, \forall 1 \leq i \leq n\}$

Distance from original image. All our attacks aim to modify an original image X into $X' = X + \delta$ such that the distance between their respective hashes is larger than a given threshold T. We denote by $f_X : \mathcal{I} \to \mathcal{O}'$ the function mapping a perturbation δ to the distance

between the hashes of $X' = X + \delta$ and X: $f_X(\delta) = d(h(X), h(X + \delta))$. For example, for real-valued hashes $\mathcal{O}' = \mathbb{R}$, while for bit-valued hashes $\mathcal{O}' \subset \mathbb{N}$ (for details see Sec. 8.5). For convenience and because each image is attacked separately, we drop the subscript and use $f := f_X$ when there is no ambiguity.

Visual similarity. We quantify the visual similarity between original image X and a modified image X' using the $\mathbb{L}_p$ norm of $X - X'$, where $\mathbb{L}_p : X \in \mathbb{R}^n \rightarrow ||X||_p = \big(\sum_{i=1}^{n} |X^i|^p\big)^{1/p}$. The Euclidean ($p = 2$), $\mathbb{L}_1$ and $\mathbb{L}_\infty$ norms are common choices in the adversarial ML literature to measure visual similarity. To compare the visual similarity between original and modified images independently of the image size, we will use the $\mathbb{L}_p$ perturbation per pixel, defined as $\mathbb{L}_{p,\text{pixel}} : X \in \mathbb{R}^n \rightarrow \big(\frac{1}{n}\sum_{i=1}^{n} |X^i|^p\big)^{1/p}$.

8.4.2 Black-box attack

Approach. Our black-box attack attempts to maximise the distance $f(\delta)$ between the hashes of original and modified images X and $X' = X + \delta$ while ensuring the perturbation $||\delta||_p$ is smaller than a fixed constant ϵ. We take this approach instead of directly minimising the perturbation under the constraint that $f(\delta) > T$ as we found it impractical to enforce the constraint on f in a black-box setting.

More specifically our attack seeks a solution to the following optimisation problem:

$$\text{Find: } \max_{\delta} \min(T, f(\delta)) \tag{8.6}$$

$$\text{s.t.: } ||\delta||_p \leq \epsilon \tag{8.7}$$

$$\delta \in \mathcal{I} - X \tag{8.8}$$

The objective is the minimum between T and $f(\delta)$. When it is equal to T (meaning that a perturbation was found such that $f(\delta) > T$), the program stops as the objective is constant with respect to δ. When it is equal to the $f(\delta)$ (meaning that $f(\delta) \leq T$), we perform gradient ascent in search for a better solution. The first constraint requires that the perturbation's $\mathbb{L}_p$ norm does not exceed ϵ. The second constraint requires that the modified image remains valid.

To ensure that the perturbation is as small as possible, we start with a very small admissible perturbation ϵ and gradually increase it when the above program fails to find a solution within a reasonable amount of steps.

Gradient estimation. We use zero-order gradient estimation via Natural Evolutionary Strategies (NES) [284, 348] to estimate the gradient of the distance function f with respect to the perturbation δ and perform gradient ascent. This is a popular technique for optimisation under black-box assumptions and has already been used to obtain adversarial perturbations against image classification models [161].

The NES-based strategy for gradient estimation can be described as a special case of estimation using finite-differences on a random Gaussian basis. We estimate the gradient of f using the following equation:

$$\nabla_\delta E[f(\delta)] \approx \frac{1}{\sigma d} \sum_{i=1}^{d'} \theta_i f(\delta + \sigma\theta_i) \tag{8.9}$$

where $\theta_i \sim N(0, I_n), 1 \leq i \leq d'$ are samples from a standard multivariate normal distribution over $\mathbb{R}^n$. Nesterov and Spokoiny [233] showed through theoretical analysis that the number of samples d' required to estimate the gradient scales linearly with the input dimension n.

Algorithm 13 details the black-box gradient estimation procedure. To reduce the variance of our estimate, we use antithetic sampling: we sample Gaussian noise θ_i for $i \in \{1, ..., \frac{d'}{2}\}$ and set $\theta_j = -\theta_{d'-j+1}$ for $j \in \{(\frac{d'}{2} + 1), ..., d'\}$ with d' an even number (line 3). This optimisation has been empirically shown to improve performance of NES [284]. To satisfy the image bound constraints for $X + \delta + \sigma\theta$ which will be given as input to the hashing function (in order to compute $f(\delta + \sigma\theta)$), we replace each noise sample with a clipped version (line 5).

Perturbation update. Algorithm 14 details the perturbation update. At each step t, we use the sign of the estimated gradient, sign($grad$) along with momentum μ to update δ_t (lines 3-4). Sign gradients have been previously used by Goodfellow et al. to develop adversarial perturbations against image classification models [141]. We clip the resulting image $X + \delta_{t+1}$ to ensure it is within the image bounds (line 5), and we enforce the visual similarity constraint on the updated δ_{t+1} by projecting it onto the ball of $\mathbb{L}_p$ norm ϵ (lines 6-7).

Algorithm 13 GRAD: Gradient estimation for $f(\delta)$

Inputs:
f: Function whose gradient is to be estimated.
X: Original image to be attacked, of size n.
δ: Point at which the gradient is to be estimated.
d': Number of Gaussian samples (should be even).
σ: Scaling factor for Gaussian samples $\sim N(0, I_n)$.

Output:
$grad$: $\nabla_\delta E[f(\delta)]$, estimate of the gradient of $f(\delta)$

Initialise:
$\theta_i \leftarrow N(0, I_n)$, for $i \in \{1, ..., \frac{d'}{2}\}$
$\theta_i \leftarrow -\theta_{d'-i+1}$, for $i \in \{(\frac{d'}{2}+1), ..., d'\}$
$grad \leftarrow 0$

for $i = 1$ to d' **do**
    $\theta'_i \leftarrow \max(\min(1, X + \delta + \sigma\theta_i), 0) - X - \delta$
    $grad \leftarrow grad + f(\delta + \theta'_i) * \theta'_i * \frac{1}{\sigma d'}$
end for

Perturbation bound. To ensure that the visual similarity is comparable across images of different sizes, the bound on the perturbation norm (ϵ) is derived from a constant independent of the image size (denoted ϵ_{norm}) for each image attacked. Perturbing each pixel by 1% in the same image but with different sizes would result in a visually similar perturbation but different values of $||\delta||_p$. In the attack, we increment ϵ_{norm} starting from small values, and the corresponding actual perturbation bound used in the attack (dependent on the image size) will be $\epsilon = \epsilon_{\text{norm}} * n^{\frac{1}{p}}$.

Making the attack more efficient. Because the number of samples d' required to estimate the gradient scales linearly with the image size [233], we attack a small-dimension resized and grayscaled version of the original image, which we denote by $\bar{X}$ (of size $n' < n$). The attack produces a perturbation $\bar{\delta}$, which we map to a final perturbation δ in the original space using Alg. 15. Our strategy, which we show to work in practice, is informed by three insights: (1) most perceptual hashing algorithms are known to be invariant to grayscaling and resizing of the image [326]. This means that, for an image X, $h(X)$ does not change much even when X is recoloured or resized. (2) both grayscaling, i.e., converting a three-channel RGB image to single-channel image, and resizing to a smaller sized image reduce the size of the original image X and (3) optimising the perturbation for the grayscale image and mapping it to a

Algorithm 14 UPDATE: Estimating δ_{t+1} via gradient ascent

Inputs:
δ_t: Perturbation after the tth iteration.
$grad_t$: $\nabla_\delta E[f(\delta_t)]$, estimate of the gradient of $f(\delta_t)$.
$prev\_grad_t$: Weighted sum of previous gradients.
X: Original image to be attacked.
ϵ: Upper bound on $||\delta||_p$.
μ: Momentum parameter.
η: Step size to update δ in each iteration.

Output:
δ_{t+1}: Perturbation after the $(t+1)^{th}$ iteration.
$prev\_grad_{t+1}$: Updated weighted sum of gradients.

$grad_t \leftarrow \mu * prev\_grad_t + (1-\mu) * grad_t$
$\delta_{t+1} \leftarrow \delta_t + \eta * \text{sign}(grad_t)$
$\delta_{t+1} \leftarrow \min(\max(X + \delta_{t+1}, 0), 1) - X$
if $||\delta_{t+1}||_p > \epsilon$ **then**
    $\delta_{t+1} \leftarrow \delta_{t+1} * \epsilon / ||\delta_{t+1}||_p$
end if
$prev\_grad_t \leftarrow grad_t$

corresponding perturbation in the RGB space leads to smoother perturbations than directly optimising for the RGB image.

Algorithm 15 details our algorithm for inverting the perturbation. Using a resize transformation, we map the single-channel perturbation $\bar{\delta}$ (of size n') to a corresponding perturbation of same single-channel size as the original image X ($n/3$), denoted by δ_{gray}. The final pixel value δ_c^i for channel c is derived from δ_{gray} such that (1) δ_c^i is proportional to X_c^i, the intensity for pixel i in the original image, and (2) the modified pixel $X_c^i + \delta_c^i$ is within the image bounds.

Complete attack. Algorithm 16 describes the complete attack for an image and X of size n and threshold T. We initialise the image-independent perturbation bound to ϵ_0. We grayscale and resize X to obtain $\bar{X}$ of smaller size n'. In each iteration t, the algorithm estimates the gradient and performs a gradient update. If the distance $f_{\bar{X}}(\bar{\delta})$ reaches a plateau as quantified by the finite difference Δf (and the number of previous iterations for which $\Delta f \leq 0$), we consider that the current ϵ is too small to allow for $f(\bar{\delta}) > T$. We thus increase the image-independent perturbation bound ϵ_{norm} by a fixed step η_ϵ and update ϵ accordingly. As soon as the distance $f_{\bar{X}}(\bar{\delta}) > T$, δ is computed from $\bar{\delta}$. The algorithm stops if $f_X(\delta) > T$ as well, in which case the attack is considered successful. If after m iterations the distance $f_X(\delta)$ does not

Algorithm 15 InverseDelta: Calculate perturbation δ for original image X from perturbation $\bar{\delta}$

Inputs:
  X: Original image to be attacked of size n.
  $\bar{\delta}$: Optimal perturbation for $\bar{X}$.
Output:
  δ: Perturbation for X.
Initialise:
  $X_c \leftarrow$ channel c of X, for $c \in \{R, G, B\}$
  $\text{mean}(X_c) \leftarrow \frac{X_R + X_G + X_B}{3}$
  $n_{gray} \leftarrow n/3$
$\delta_{gray} = \text{resize}(\bar{\delta}, n_{gray})$
for c in $\{R, G, B\}$ **do**
  for $i = 1$ to n_{gray} **do**
    if $\delta^i_{gray} \leq 0$ **then**
      $\delta^i_c = \delta^i_{gray} \frac{X^i_c}{\text{mean}(X^i_c)}$
    else
      $\delta^i_c = \delta^i_{gray} \frac{1 - X^i_c}{1 - \text{mean}(X^i_c)}$
    end if
    $\delta^i_c = min(max(X^i_c + \delta^i_c, 0), 1) - X^i_c$
  end for
end for

exceed T, the attack is considered unsuccessful.

Diversity of perturbations. For the attack to be resistant against simple mitigation strategies like expanding the database with hashes of the modified image X', the attack should generate multiple random perturbations for an image, resulting in different and unpredictable output hashes. To achieve this objective, we make two modifications to our black-box attack. First, we initialise our perturbation $\bar{\delta}_0$ to be a valid random non-zero perturbation such that $||\bar{\delta}_0||_p = \epsilon_{start} * n'^{\frac{1}{p}}$ for some small value of ϵ_{start}. More specifically, we uniformly sample $\bar{\delta}_0$ from a set of valid perturbations, scale it to have an $\mathbb{L}_p$-norm of ϵ_{start} and clip to limit it within bounds.

$$\bar{\delta}_0 \sim U(-\bar{X}, 1 - \bar{X}) \tag{8.10}$$

$$\bar{\delta}_0 = \epsilon_{start} * n'^{\frac{1}{p}} * \frac{\bar{\delta}_0}{||\bar{\delta}_0||_p} \tag{8.11}$$

$$\bar{\delta}_0 = \min(\max(\bar{\delta}_0, 0), 1) \tag{8.12}$$

Algorithm 16 Black-box attack against perceptual hashing algorithms

Inputs:
X: Original image to be attacked of size n.
h: Perceptual hashing algorithm to be attacked.
d: Distance metric corresponding to perceptual hashing algorithm h.
T: Threshold to be attacked.
$n'(< n)$: Size of the resized and grayscaled image $\bar{X}$
ϵ_0: Starting value for the maximum perturbation allowed.
η_ϵ: Step size for ϵ.
k: Number of previous iterations used to detect plateauing of $f_{\bar{X}}$.
μ, η: Parameters required for updating the gradient.
d', σ: Number of samples and scale for Gaussian samples used in gradient estimation.
s: Seed defined by the adversary.
m: Maximum number of iterations.
Output:
δ: Perturbation for original image X.
Initialise:
$\bar{X} \leftarrow \text{resize}(\text{rgb2gray}(X), n')$
$\bar{\delta}_0 \leftarrow 0$
$prev\_grad_0 \leftarrow 0$
$seed \leftarrow (X, s)$
$\epsilon_{norm} \leftarrow \epsilon_0$
$\epsilon \leftarrow \epsilon_{norm} * n^{\frac{1}{p}}$
setseed($seed$)
for $t = 0$ to $m - 1$ **do**
  $grad_t \leftarrow \textsc{Grad}(f_{\overline{X}}, \bar{X}, \bar{\delta}_t, d', \sigma)$
  $\bar{\delta}_{t+1}, prev\_grad_{t+1} \leftarrow \textsc{Update}(\bar{\delta}_t, grad_t, prev\_grad_t, \overline{X}, \epsilon, \mu, \eta)$
  $\Delta f_t \leftarrow f_{\overline{X}}(\bar{\delta}_{t+1}) - f_{\overline{X}}(\bar{\delta}_t)$
  if $\text{Count}([\Delta f_{t-k+1}, ..., \Delta f_t] \leq 0) > \frac{k}{2}$ **then**
    $\epsilon_{norm} \leftarrow \epsilon_{norm} + \eta_\epsilon$
    $\epsilon \leftarrow \epsilon_{norm} * n'^{\frac{1}{p}}$
  end if
  if $f_{\overline{X}}(\bar{\delta}_{t+1}) > T$ **then**
    $\delta = \textsc{InverseDelta}(X, \bar{\delta}_{t+1})$
    if $f_X(\delta) > T$ **then**
      break
    end if
  end if
end for

Second, we observe that requiring the perturbation to be smaller than the constant used in the default black-box attack can lead to less diverse outputs, even when different starting points are used in the optimisation. So we allow for a larger maximum perturbation.

The two algorithm modifications lead to more diverse hashes, at a small cost to the visual similarity.

Reproducibility. In order to ensure our attack results are reproducible, we initialise the attack using both the image X and the adversary-defined seed s (line 4 in Algorithm 16). This ensures that the directions θ being probed for gradient estimation are not the same for all images even when the adversary-defined seed s remains the same. For diversity, using different adversary-defined seeds leads to diverse perturbations for the same image. An adversary can either manually define the seed, or set it based on processor clock to generate a diverse perturbation everytime.

8.4.3 White-box attacks for DCT-based hashes

We develop a principled attack to devise a minimum perturbation for DCT-based perceptual hashing algorithms. The Discrete Cosine Transform (DCT) [12] is a very popular image compression algorithm. DCT maps a discrete signal to linear combinations of the original signal and cosines of different frequencies. The signal can then be compressed by noting that coefficients corresponding to lower frequencies encode the most important features. This property makes DCT suitable for JPEG image compression [344] and several perceptual hashing algorithms like pHash [78] and Facebook's PDQ [86].

Attack intuition. Our attack exploits the linearity and orthogonality of DCT. By using the Euclidean distance to measure both the input perturbation and the distance between original and modified hashes, we show that minimal perturbations can be found as linear combinations of eigenvectors derived from the linear transform. We further show that the minimal perturbation needed to exceed the threshold T is equal to T exactly.

Overview of DCT-based perceptual hashing. DCT-based perceptual hashing algorithms apply a set of transformations to the original image such as conversion to grayscale, resizing, or blurring, which results in a smaller $k \times k$ image where k is a parameter. The DCT algorithm takes as input the $k \times k$ image and returns an output of the same size, whose dimensionality is

then reduced by keeping a submatrix of pixels and discarding the rest:

- pHash uses a parameter $k = 32$ and keeps the 8×8 submatrix of pixels from the intersection of rows and columns 2 to 9.
- PDQ uses $k = 64$ and keeps the 16×16 submatrix of pixels from the intersection of rows and columns 2 to 17.

The DCT transform applies the following operations to an image of size $k \times k$:

$$h_{DCT} : X \longrightarrow MXM^T, \text{ with } M \in \mathbb{R}^{k \times k}, M_{ij} = \sqrt{\frac{2}{k}} \Lambda_i \cos\left[\frac{\pi}{k}\left(j + \frac{1}{2}\right)i\right], 1 \leq i, j \leq k \quad (8.13)$$

$$\Lambda_i = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } i = 0 \\ 1, & \text{otherwise} \end{cases} \quad (8.14)$$

The matrix M is orthogonal, i.e., $MM^T = I_k$, where I_k denotes the identity matrix of size $k \times k$.

The submatrix extraction step selecting the pixels from rows a to b and columns a to b can be written as $h'_{DCT} : X \to (MXM^T)_{a:b,a:b}$. The parameters a and b ($1 \leq a \leq b \leq k$) are chosen when developing a perceptual hashing algorithm: a is typically small to extract lower frequencies while b controls the hash size. We rewrite the mapping as $h'_{DCT} : X \to M'XM'^T$, with $M' = M_{a:b,1:k} \in \mathbb{R}^{(b-a+1) \times k}$, i.e., M' is the matrix obtained by extracting rows a to b from M. Similarly to above, $M'M'^T = I_{b-a+1}$. For more conciseness we set from now on $c = b-a+1$.

Linearity of DCT. It is straightforward to show that h'_{DCT} is a linear transformation mapping a $k \times k$-dimensional input to a $c \times c$-dimensional output. Simplifying the notation, h'_{DCT} can be rewritten as the linear transform of a vector (the flattened input matrix X):

$$h'_{DCT} : \begin{cases} \mathcal{I} \longrightarrow \mathbb{R}^{c^2} \\ X \longrightarrow AX \end{cases} \quad (8.15)$$

$$\text{with } A \in \mathbb{R}^{c^2 \times k^2} \tag{8.16}$$

$$A_{c_1 \times c + c_2, k_1 \times k + k_2} = M'_{1+c_1,1+k_1} \times M'_{1+c_2,1+k_2}, \text{ for } 0 \leq c_1, c_2 \leq c-1, 0 \leq k_1, k_2 \leq k-1 \tag{8.17}$$

Proposition 3. *A is an orthogonal matrix, i.e., $AA^T = I_{c^2}$.*

Proof. Let $c = c_1 \times c + c_2$ and $c' = c'_1 \times c + c'_2$. We develop $(AA^T)_{c,c'}$ as follows:

$$(AA^T)_{c,c'} = \sum_{k_1=0}^{k-1} \sum_{k_2=0}^{k-1} A_{c_1 \times c + c_2, k_1 \times k + k_2} \times A_{c'_1 \times c + c'_2, k_1 \times k + k_2} \tag{8.18}$$

$$= \sum_{k_1=0}^{k-1} \sum_{k_2=0}^{k-1} M'_{1+c_1,1+k_1} \times M'_{1+c_2,1+k_2} \times M'_{1+c'_1,1+k_1} \times M'_{1+c'_2,1+k_2} \tag{8.19}$$

$$= (\sum_{k_1=0}^{k-1} M'_{1+c_1,1+k_1} \times M'_{1+c'_1,1+k_1})(\sum_{k_2=0}^{k-1} M'_{1+c_2,1+k_2} \times M'_{1+c'_2,1+k_2}) \tag{8.20}$$

$$= (M'M'^T)_{1+c_1,1+c_2} \times (M'M'^T)_{1+c'_1,1+c'_2} \tag{8.21}$$

Because $M'M'^T = I_c$, it follows that $(AA^T)_{c,c'} = 1$ if $c = c'$ and 0 otherwise. Indeed, $c = c'$ if and only if $c_1 = c'_1$ and $c_2 = c'_2$, meaning that the terms being multiplied in the last equality can both be non-zero only when $c = c'$, in which case they are both equal to 1. □

Proposition 4. *Let X and δ be two $k \times k$ real matrices. Then $f(\delta) = d(h(X), h(X+\delta)) \geq T$ (with d the Euclidean distance) implies that $||\delta||_2 \geq T$.*

Proof. We can write $f_{DCT}(\delta) = ||h_{DCT}(X) - h_{DCT}(X+\delta)||_2 = ||A\delta||_2 = \sqrt{\delta^T A^T A \delta}$. Since $A^T A$ is symmetric, it is diagonalizable in a real orthonormal basis of eigenvectors $(\delta_i)_{1 \leq i \leq k^2}$ by the spectral theorem. The eigenvalues are non-negative because $A^T A$ is positive semidefinite. Since $AA^T = I_{c^2}$, it follows that the eigenvalues can only be 1 or 0. By Proposition 5 below, the multiplicities of eigenvalues are c^2 for 1 and $k^2 - c^2$ for 0. Without loss of generality, we reorder the eigenvectors by decreasing eigenvalue. Finally, if $\delta = \sum_{i=1}^{k^2} \alpha_i \delta_i$, we can write:

$$f(\delta) = \sqrt{\sum_{i=1}^{c^2} \alpha_i^2} \leq \sqrt{\sum_{i=1}^{k^2} \alpha_i^2} = ||\delta||_2 \tag{8.22}$$

Equality holds if and only if the perturbation δ belongs to the vector space spanned by eigenvectors $(\delta_i)_{1 \leq i \leq c^2}$. □

Proposition 5. *The eigenvalues of A^TA are 1 with multiplicity c^2 and 0 with multiplicity $k^2 - c^2$.*

Proof. Let λ be an eigenvalue of A^TA and $x \neq 0$ such that $A^TAx = \lambda x$. We multiply by A to the left and obtain $\lambda Ax = A(A^TAx) = (AA^T)Ax = Ax$. We distinguish two cases: either $Ax = 0$, which implies that $\lambda = 0$ or $Ax \neq 0$, which implies that $\lambda = 1$. It follows that the eigenvalues of AA^T can only be 0 or 1.

We denote by $m(\lambda)$ the multiplicity of eigenvalue λ. From the above analysis, $dim(Ker(A)) \leq m(0)$ and $rank(A) \leq m(1)$. By summation we obtain $k^2 \leq m(0) + m(1) = k^2$, therefore $m(1) = rank(A)$.

We show that the rank of A is c^2. Indeed $rank(A) \leq min(c^2, k^2) = c^2$. If it were the case that $rank(A) < c^2$, then there would be $x \in \mathbb{R}^{c^2}$ such that $x^TA = 0$ but $x \neq 0$. Since $AA^T = I_{c^2}$, by multiplying with A^T on the right we obtain $x^T = 0$, a contradiction.

We thus conclude that $m(1) = c^2$ and $m(0) = k^2 - c^2$. □

We provide two practical ways to obtain perturbations δ such that $f(\delta) \geq T$.

Attack as an optimisation program. We frame the attack as the following non-convex optimisation program:

$$\begin{aligned} &\text{Maximise} : ||A\delta||_2^2 \\ &\text{s. t.} : ||\delta||_2^2 \leq T^2 \\ &\delta \in \mathcal{I} - X \end{aligned} \tag{8.23}$$

The first constraint requires that the $\mathbb{L}_2$ norm of the input perturbation is no larger than T. The second constraint is linear and requires that the perturbed image is valid. We use the Disciplined Convex Concave Programming toolbox [300] to find a solution.

Attack using sampling. This attack samples valid perturbations of norm T *uniformly at*

Algorithm 17 White-box attack using rejection sampling

1: **Inputs:**
T: Threshold to be attacked.
$(\delta_i)_{1\leq i\leq c^2}$: Orthonormal eigenvectors of A^TA for eigenvalue 1.
2: **Output:**
δ: Minimal perturbation (of $\mathbb{L}_2$ norm equal to T) such that $f_{DCT}(\delta) = T$.
3: **while** $\delta \notin \mathcal{I} - X$ **do**
4: $\alpha \sim N(0, \mathcal{I}_c)$
5: $\alpha \leftarrow T\frac{\alpha}{||\alpha||_2}$
6: $\delta \leftarrow \sum_{i=1}^{c^2} \alpha_i\delta_i$
7: **end while**

random, using rejection sampling. Algorithm 17 describes our procedure to sample perturbations of norm T along eigenvectors that are not canceled by the linear mapping $(\delta_i)_{1\leq i\leq c^2}$ until a valid perturbation is found meaning that $\delta \in \mathcal{I} - X$. Given a threshold T, we sample uniformly at random a vector α on the $\mathbb{L}_2$-sphere of radius T in c^2 dimensions (lines 5-6). We set the perturbation to be $\delta = \sum_{i=1}^{c^2} \alpha_i\delta_i$ (line 7). Due to the orthonormality of the eigenvector basis, indeed $||\delta||_2^2 = \sum_{i=1}^{c} \alpha_i^2 = T^2$.

8.5 Experimental setup

In this section, we describe the perceptual hashing algorithms used to instantiate our attack and our attack parameters.

8.5.1 Perceptual hashing algorithms

The perceptual hashing algorithms evaluated in this work are pHash [78], aHash, dHash [1], and Facebook's PDQ [178]. They are commonly used algorithms for image deduplication or retrieval.

pHash, aHash and dHash outputs a 64-bit hash while PDQ outputs a 256-bit hash for any input image. The Hamming distance is used to compare the outputs of the hashing algorithms. Each

[1] https://hackerfactor.com/blog/index.php%3F/archives/432-Looks-Like-It.html

algorithm applies a sequence of transformations such as grayscaling and resizing followed by image feature extraction (e.g. using DCT), and finally a bit discretisation step. To experiment with an algorithm with real-valued outputs (of size 64), we remove the discretisation step of the pHash algorithm to obtain another algorithm, which we call pHash continuous. The Euclidean distance is used to compare the outputs of pHash continuous.

pHash and pHash continuous apply grayscaling, box blurring and resizing resulting in a 32×32 image. They then apply the DCT as explained in Sec. 8.4.3, with parameters $a = 1, b = 8, k = 32$ to get a 64-sized vector. We use this real-valued vector as the output of pHash continuous. pHash assigns bits larger than the median to 1 and 0 otherwise to obtain the final hash.

dHash and aHash apply grayscaling followed by resizing to get a 9×8 and 8×8 image, respectively. aHash outputs a 64-bit hash by flattening the image, and setting the pixels above mean value to 1 and 0 otherwise. dHash computes a difference between consecutive columns of the image, flattens the computed difference, sets the values greater than 0 to 1, and 0 otherwise to report a 64-bit hash.

PDQ applies grayscaling to the image followed by a series of transformations (see [178] for details) to get a 64×64 image. It then applies the DCT algorithm as explained in Sec. 8.4.3 with $a = 1, b = 16, k = 64$, followed by bit quantisation by setting bits larger than the median to 1, and 0 otherwise to finally obtain a 256-bit hash.

8.5.2 Attack parameters

Black-box attack. We use the following parameters to run our black-box attack. All the images are converted to grayscale and resized to $n' = 64 \times 64 = 4096$ dimensions. We use the $\mathbb{L}_2$ norm to quantify visual similarity and $\epsilon_0 = \eta_\epsilon = 1/255$. We detect the plateauing behaviour of the objective function, i.e., when it stops increasing steadily, based on the last $k = 10$ iterations. To estimate the gradient, we use a number of Gaussian samples equal to $d' = 800$. We run the attack for a maximum number of iterations $m = 10,000$, although in practice only a few hundred iterations are required for most images. We use a momentum $\mu = 0$

for all hashing algorithms but pHash continuous for which we use $\mu = 0.8$. We set the scale of the Gaussian noise σ and the learning rate η to: $\sigma = 0.001, \eta = 0.001$ for pHash continuous, $\sigma = 0.1, \eta = 0.01$ for pHash, $\sigma = 0.1, \eta = 0.001$ for aHash, $\sigma = 0.1, \eta = 0.001$ for dHash and $\sigma = 0.1, \eta = 0.01$ for PDQ. An adversary might want to adjust these parameters for each image, e.g., in order to run the attack more efficiently. For simplicity, we here use the same parameters for all the images. We chose the values of these parameters for each perceptual hashing algorithm based on early experiments on a handful of images from the Stanford Dogs dataset [179].

Diversity. In the diversity experiments, we use the algorithm with modifications for diversity as detailed in Sec. 8.4.2. We set $\epsilon_0 = 0.25$ and $\eta_0 = 0.01$. Our results on $\mathbb{L}_2$ perturbations per pixel for different algorithms and different thresholds in Fig. 8.3 show that pHash continuous requires only a small perturbation to generate successful perturbations, while for the other algorithms we need a threshold-specific ϵ_{start}. With these observations, we set $\epsilon_{start} = 0$ for pHash continuous for all thresholds. For aHash and dHash we set $\epsilon_{start} = 0.03, 0.04, 0.05$, and for pHash and PDQ we set $\epsilon_{start} = 0.06, 0.08, 0.1$ for three respective thresholds.

Implementation details. We implement the framework and the perceptual hashing algorithms in Python 3.7 [266] using NumPy [147], OpenCV [45], and SciPy [341]. We further use DCCP [300] for white-box optimisation and Faiss [171] for efficiently matching hashes against the database.

8.5.3 Dataset

We use the ImageNet dataset from ILSVRC 2012 challenge [280] to empirically evaluate our attacks. More specifically, we use the subset (train, validation, and test combined) obtained after removing, for privacy reasons, all the images predicted by Yang et al. [351] to contain human faces. We also remove duplicates as described below, leaving us with a dataset of 1179448 images.

To correctly estimate the false positive rate (FPR) of client-side scanning at low thresholds, it

is necessary to remove duplicate images between the database and the images being matched. Although the website of ImageNet claims duplicates were removed, a manual inspection revealed that this was not the case. The dataset contains both exact duplicates (the same images with no or a few modified pixels) and near-duplicates, where one image is an edited copy of the other [104] or clearly taken within a few seconds from the other.

To detect exact and near-duplicates, we use a semi-automated procedure based on standard perceptual hashing algorithms. We first flag potential duplicates using perceptual hashing algorithms, then manually label them as either true or false duplicates, and finally remove the true duplicates. In total, 8526 images duplicates were removed from the dataset.

Detecting duplicates. We start by retrieving for each image their matches at distance d calculated using PDQ. To ensure that we do not overlook duplicates that could potentially result in overestimated false positive rates, we iterated over all the distances d between 0 and 70, the lowest threshold attacked for PDQ (see Table 8.1). We traverse ImageNet in decreasing image size order, to ensure that potential duplicates are lower-quality images. We manually label each duplicate as either "true" or "false" and only remove the true duplicates. We acknowledge that the notion of "true" duplicate can be subjective, especially for near-duplicates. Consequently, all edge cases were labeled by the three authors.

Figure 8.1 shows the number of potential duplicates detected for each distance d by PDQ (for clarity, we do not display the value for $d = 0$ for which 7136 are detected). As it is unfeasible to inspect all of the images at $d = 0$, we inspected a sample of 100 images with duplicates and found all of them to be true (exact) duplicates. For d ranging from 2 to 54, we found a total of 1404 matches which we manually inspected. We found most of them (1339) to be true duplicates, as shown on Figure 8.1. For d ranging from 56 to 66, we found a total of 842 potential duplicate pairs. We manually inspect them and found 151 of them to be true duplicates which we remove. The larger thresholds ($d = 68$ and $d = 70$) contained more than 400 duplicate pairs. A manual inspection of a random subset revealed that a large majority of them (93%, resp. 96%) are false duplicates. Considering the large number of images that would have to be manually inspected for a likely minimal impact we did not remove further images.

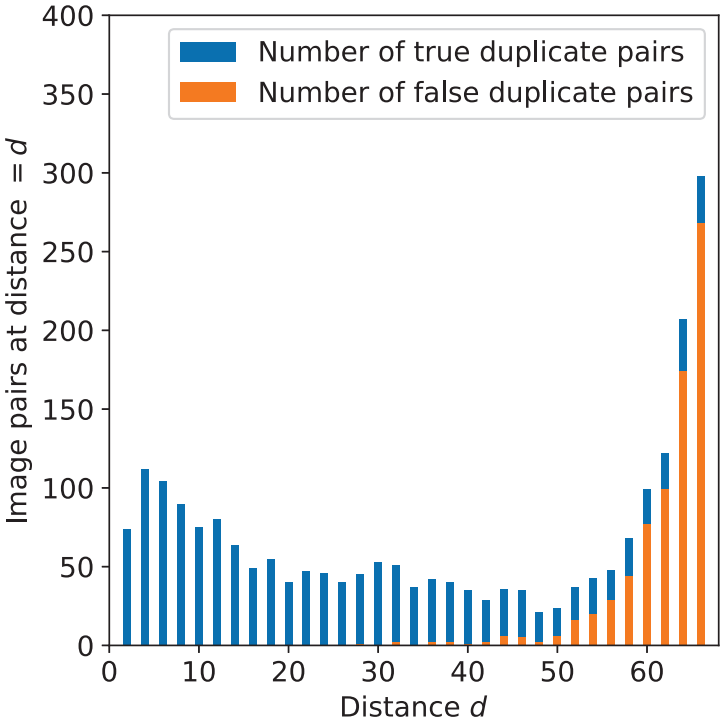


Figure 8.1: **Number of potential duplicates detected for each distance d by the PDQ algorithm.**

As the number of false duplicates is already very large, we believe that the small remaining fraction of duplicates is unlikely to significantly impact the FPR of the client side scanning system.

Other hashing algorithms. To avoid potential biases, we also check for duplicates using the other hashing algorithms, in the order given below. More specifically, we retrieve matches at distance d ranging from 0 to the lowest threshold attacked in this work for that algorithm:

1. For pHash, 15 ($d = 0$) and 69 ($d = 2$) images are flagged as potential duplicates. Out of these, 24 are true duplicates that we remove. For $d = 4$ and $d = 6$, as more than 400 image pairs would have to be inspected (namely 489 and 2950), we sample 100 random images with duplicates and find 99% and 97% of them to be false duplicates.

2. For pHash continuous, we retrieve all images with potential duplicates within distance intervals [0, 0.1), [0.1, 0.2), up to [0.5, 0.6). We retrieved 4 matches within the first interval, none of which was a true duplicate. We retrieved 519 matches within the second interval and, upon manual inspection of 100 pairs, we found no true duplicate. For the

remaining intervals, we retrieved 5K, 2K, 42K, and 79K. We manually inspected 100 pairs in each of these and found no true duplicate. In fact, in many cases, a large number of visually very different matches were retrieved, indicating that the thresholds ranges were already very high for duplicate detection.

3. For aHash, we found a very high number of matches, e.g., 59K for $d = 0$ and 140K for $d = 3$, the lowest threshold attacked in this work. We inspected 500 pairs for $d = 0$ and found only one true duplicate.

4. For dHash, we find 62 matches for $d = 0$ and 191 for $d = 1$. We manually inspect them and find that only one of these is a true duplicate which we remove. We then find 677 matches for $d = 2$, 2125 matches for $d = 3$, and 6184 matches for $d = 4$, the lowest threshold attacked. We inspected 100 pairs for each threshold and found no true duplicate.

8.6 Results

We sample from the dataset two mutually exclusive sets of images, of sizes N and M, uniformly without replacement. The images in the first set are used to create the database $\mathcal{D}$ of size N (by default, we use $N = 100$K images) and the images in the second set are used as images visually different to those in the database, to estimate the false positive rate (we use $M = 500$K images). We select three thresholds for each perceptual hashing algorithm, shown in Table 8.1 (see Discussion for an analysis). We run the black-box attack on $N' = 1,000$ randomly sampled images from the database $\mathcal{D}$. While the threshold will vary depending on the use case, the lowest threshold would be typically used in practice (see e.g., [178] for PDQ). The second and in particular the third threshold would probably generate too many false positives in practice. We include them to test our attack in more challenging scenarios. Given a threshold T and N' images to be attacked, we compute the *attack effectiveness* as the proportion of images that are successfully attacked, i.e., a perturbation is found satisfying $f_X(\delta) > T$. We repeat the attack $R = 10$ times.

Algorithm	Thresholds
pHash continuous	0.6, 1.1, 1.6
pHash	6, 10, 14
aHash	3, 5, 7
dHash	4, 9, 12
PDQ	70, 85, 90

Table 8.1: **Thresholds selected to simulate the client-side scanning system.**

Our black-box attack manages to successfully attack all images ($N' \times R = 10000$) for all perceptual hashing algorithms but one, reaching an attack effectiveness of 99.9%. More specifically, we found that one image could not be attacked at the thresholds considered when using aHash for one particular instantiation of our attack. All other images were all successfully attacked for all perceptual hashing algorithms and thresholds considered.

Fig. 8.2 shows examples of images successfully perturbed using our attack along with the threshold used and resulting $\mathbb{L}_2$ perturbation per pixel. The modified image always preserves the visual content of the original image with PDQ seemingly requiring the most visually perceptible modifications. Values of up to $\mathbb{L}_{2,\text{pixel}} \approx 0.10$ resulting in very small changes are typically needed for most images even with the largest thresholds considered. We chose images where the subject is in focus so that the effect of perturbations can be clearly seen and we refer the reader to Fig. 8.13- 8.17 at the end of the chapter for the complete results of our attack against these images for all perceptual hashing algorithms and thresholds.

Fig. 8.3a shows that a small $\mathbb{L}_2$ perturbation per pixel is enough to successfully attack most of the images for all hashing algorithms. We obtain similar results using the popular Learned Perceptual Image Patch Similarity (LPIPS) distance [363] (see Fig. 8.3b) that measures the perceptual distance between images using intermediate layers of a pre-trained image classification model. We can see that the perturbation increases overall with the threshold for all perceptual hashing algorithms. This is expected as a larger perturbation might be needed to push an image at least T away from the original. However, even for the largest thresholds, we observe that an $\mathbb{L}_2$ perturbation per pixel of 0.10 is enough to successfully attack with imperceptible modification 95.3% of images for dHash (respectively 100.0%, 83.5%, 86.8% and 73.4%

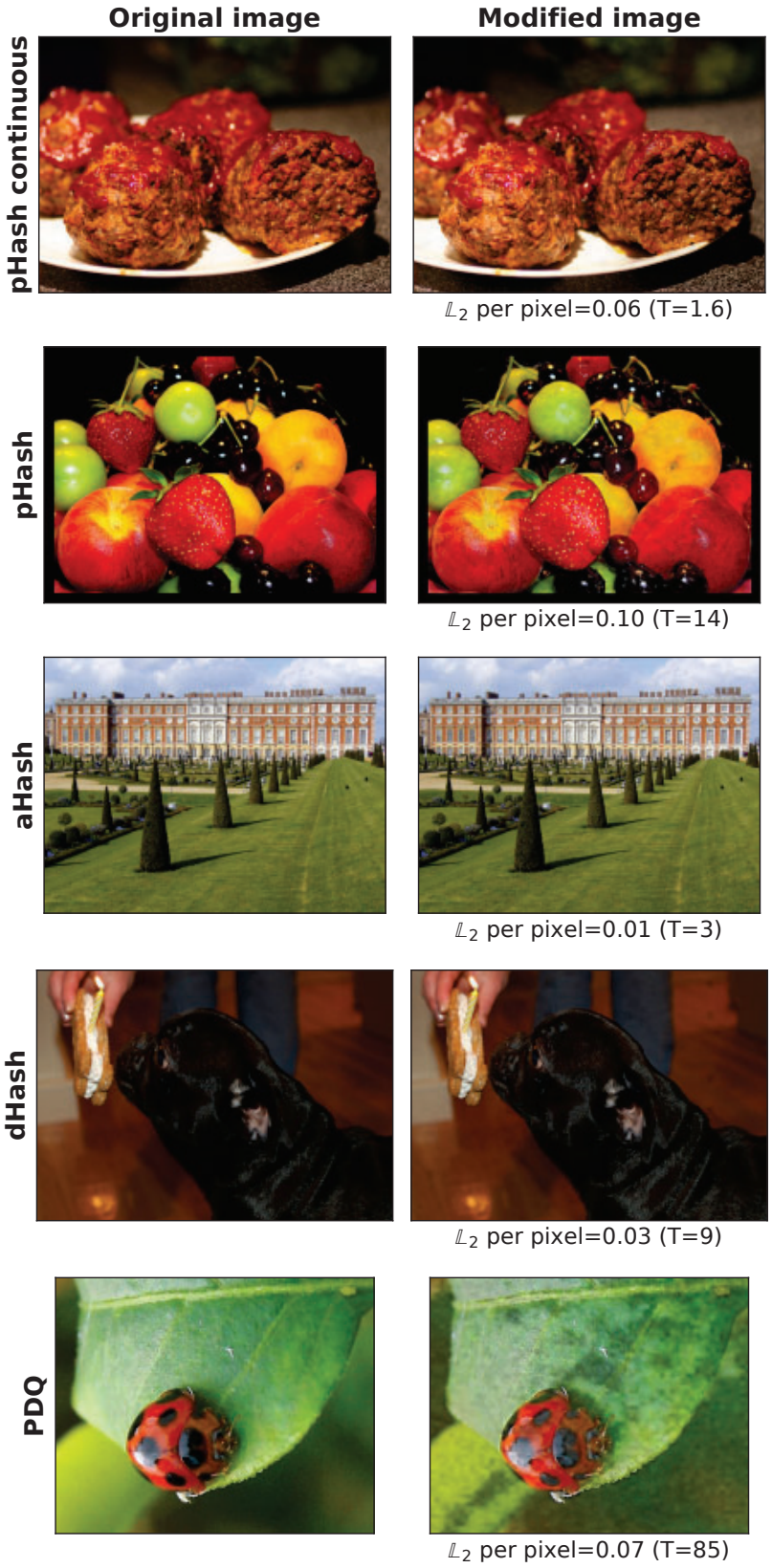
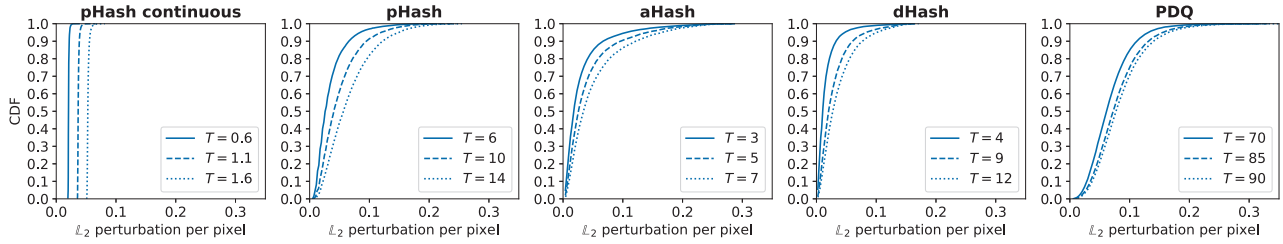
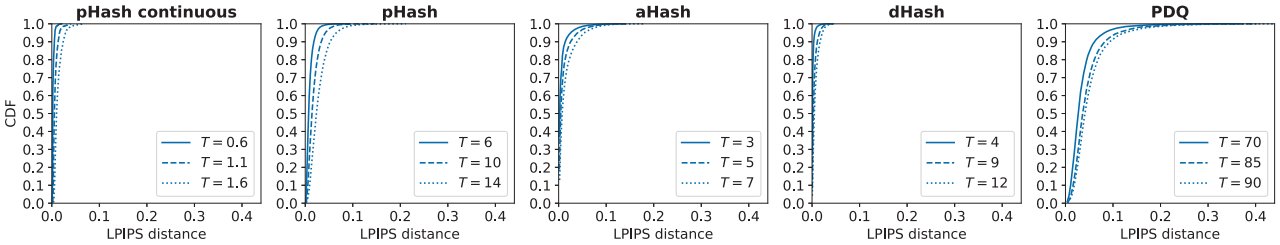


Figure 8.2: **Examples of original and modified images for different hashing functions.** All four modified images evade detection for the threshold T while maintaining a small $\mathbb{L}_2$ perturbation per pixel and high visual similarity to the original image.



(a) $\mathbb{L}_2$ perturbation per pixel



(b) LPIPS distance

Figure 8.3: **Visual similarity between original and modified images.** We compute the cumulative distribution function (CDF) of the (a) $\mathbb{L}_2$ perturbation per pixel and (b) LPIPS distance for different algorithms and thresholds and all successfully attacked images over 10 experiments. A lower perturbation or distance indicates a higher visual similarity between the modified and original images. The perturbation increases slowly with the threshold, but remains small in all cases.

for pHash continuous, pHash, aHash and PDQ). While results cannot be directly compared across algorithms as thresholds are algorithm-specific, we observe interesting difference in the shape of CDFs from various algorithms. pHash continuous seems to requires a similar amount of perturbation for all images (at a given threshold) with, e.g., $\mathbb{L}_2$ for $T = 1.6$ only ranging between 0.050 and 0.082. pHash, dHash, and aHash have similar distributions while PDQ seems to always require some changes to the image (resulting in a CDF shifted to the right). We hypothesise that the behaviour we observe for pHash continuous is due to two factors: (1) the attack mostly targets the DCT step where it finds an almost optimal $\mathbb{L}_2$ perturbation of $T = 1.6$ for an image size of 32×32, yielding an $\mathbb{L}_2$ perturbation per pixel of $\approx 1.6/32 = 0.05$ and (2) the other transformations from original to resized images (and from resized to original image to invert the perturbation) roughly preserve the $\mathbb{L}_2$ perturbation per pixel.

Fig. 8.4 shows that a detection system would have a very large False Positive Rate (FPR), even for the lowest threshold considered. We here empirically estimate FPR as the fraction of images among the M images outside the database flagged by the system. Even for the smallest

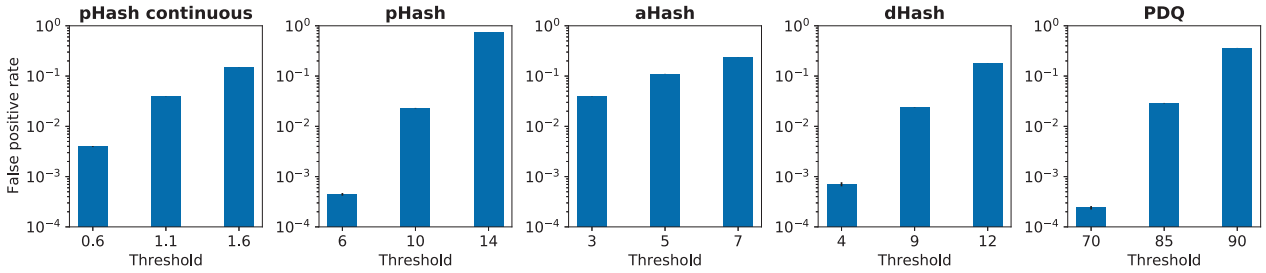


Figure 8.4: **False Positive Rate (FPR)** for different algorithms and thresholds. We use a database size of $N = 100$K and estimate the FPR by matching $M = 500$K images against the database. We show the mean with error bars for the standard deviation (which are very small) over 10 repetitions.

Hash	Estimate of the number of wrongly detected images daily / Threshold T		
pHash continuous	$\sim$ 18M ($T = 0.6$)	$\sim$ 179M ($T = 1.1$)	$\sim$ 660M ($T = 1.6$)
pHash	$\sim$ 2M ($T = 6$)	$\sim$ 104M ($T = 10$)	$\sim$ 3.3B ($T = 14$)
aHash	$\sim$ 175M ($T = 3$)	$\sim$ 496M ($T = 5$)	$\sim$ 1.1B ($T = 7$)
dHash	$\sim$ 3M ($T = 4$)	$\sim$ 106M ($T = 9$)	$\sim$ 796M ($T = 12$)
PDQ	$\sim$ 1M ($T = 70$)	$\sim$ 130M ($T = 85$)	$\sim$ 1.6B ($T = 90$)

Table 8.2: **Estimate of the number of images that would be wrongly detected daily on WhatsApp for different perceptual hashing algorithms and thresholds.** We here consider a database size of $N = 100$K, 4.5B images being shared daily [36], and a prevalence rate of illegal content of 10^{-4}.

threshold considered, the false positive rate is larger than 0.02%. While seemingly small, such values of FPR would, in practice, result in a large number of images being detected by the system and having e.g. to be sent unencrypted to be analysed. In 2017, 4.5B images were shared daily on Whatsapp alone [36]. For a prevalence rate of illegal content of $p = 10^{-4}$ and a database size of $N = 100$K, this would result in >1M false positive images having to be shared unencrypted daily (see Table 8.2), raising very significant privacy concerns. For the largest thresholds, the false positive rate increases to rates between 14.7% for pHash continuous to 73.0% for pHash resulting in hundreds of millions of images being incorrectly flagged and shared daily. We obtain very similar values for a smaller prevalence rate of $10^{-}7$.

Diversity. Contrary to the ML case, ensuring that an attack generates diverse enough perturbations every time it is run is essential to make it a practical concern for client-side scanning. Deterministic attacks could indeed easily be thwarted by expanding the database to include the original images and its modifications. To assess the ability of our attack to produce different

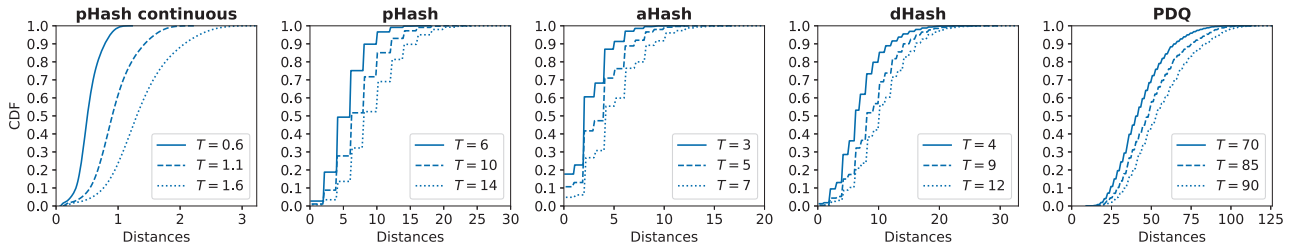


Figure 8.5: **Pairwise distances between hashes of the multiple modified images generated for the same image, for different algorithms and thresholds.** We generate $D = 50$ modified images for each ($N' = 100$) original image.

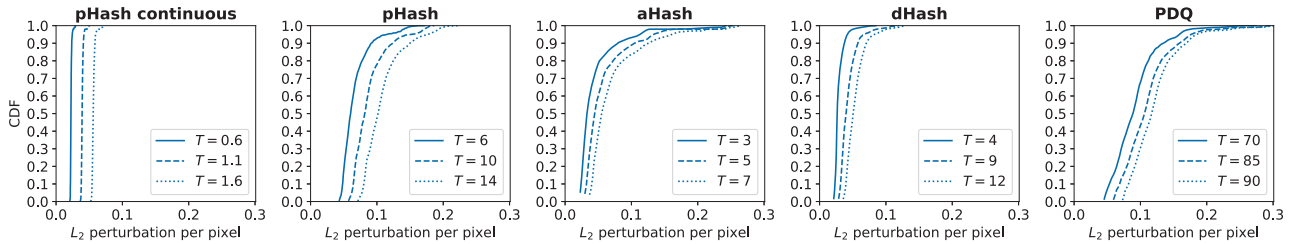


Figure 8.6: **Cumulative distribution function (CDF) of the $\mathbb{L}_2$ perturbations per pixel of the multiple perturbations generated for the same image for different algorithms and threshold.** The number of perturbations generated for each image is $D = 50$ and the number of images attacked is $N' = 100$.

perturbations, we attack $N'' = 100$ images sampled uniformly at random without replacement from the database. Each image is attacked $D = 50$ times, using the diversity-focused version of the attack and different random initialisations (seeds) (see Sec. 8.4.2). Fig. 8.6 shows the $\mathbb{L}_2$ perturbation per pixel is slightly larger than in the default black-box attack. Yet, most perturbations are imperceptible and have $\mathbb{L}_2$ perturbation per pixel ≤ 0.15 for all thresholds and algorithms.

Fig. 8.5 shows the pairwise distances between hashes of modified images generated from the same image. The distances between modified images are roughly similar to the threshold used for all the perceptual hashing algorithms evaluated. This suggests that the modified images are well distributed around the ball centred on the original image and not clustered in one part of the space. This also suggests that simple mitigation strategies like expanding the database would not be sufficient to counter the attack.

Finally, we assess the capacity of the black-box approach to produce optimal perturbations against the DCT step. We compare the results of the black-box attack to those of the white-

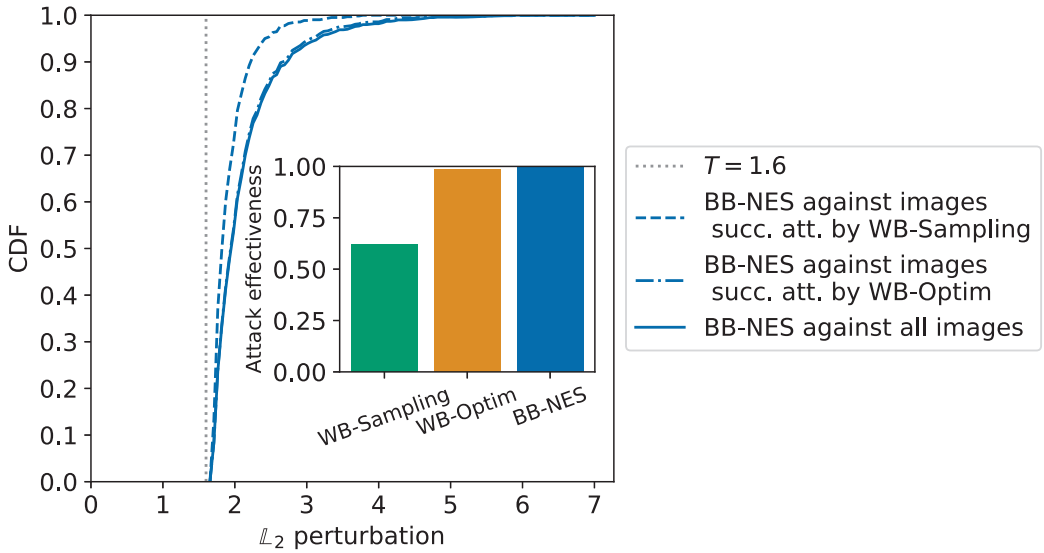


Figure 8.7: **Optimality analysis of the black-box approach**. We compute the cumulative distribution function (CDF) for the $\mathbb{L}_2$ norm of perturbations using our black-box attack (BB-NES) against the DCT step and a threshold $T = 1.6$ (gray dotted line) for $N' = 1000$ images. The inset shows the attack effectiveness for the three attacks.

box attacks and to our theoretical lower-bound (for a threshold T and the Euclidean distance, the $\mathbb{L}_2$-norm $\geq T$, see Sec. 8.4.3). We attack $N' = 1000$ images randomly sampled from ImageNet after resizing them to $k \times k$ with $k = 32$ and use the black-box attack parameters from the DCT-based pHash continuous. For the WB-Sampling approach, we generate up to 1M samples and we consider the attack successful if $\delta \in I - X$ (see Algorithm 17 for details). For the WB-Optim approach, we run the optimisation program for a maximum of 1000 iterations and for up to 10 times, until a solution is found; if none is found, the attack is considered unsuccessful. Fig. 8.7 shows that the black-box attack achieves an input perturbation no larger than 2 times the theoretical limit 96% of the time and no more than 0.5 above the limit 74% of the time. Interestingly, in this experiment, while the black-box attack always succeeds in finding a perturbation, the white-box approaches are less successful. WB-Optim and WB-Sampling achieve an attack effectiveness of only 98.4% and 62.2% respectively. We believe that WB-Sampling is less successful because the sampled perturbations are more likely to go out of the image bounds for some images, resulting in no successful perturbation. Finally, we note that the $\mathbb{L}_2$ perturbation is smaller for the images successfully attacked by WB-Sampling, perhaps because it easier to devise small perturbations for these images.

8.7 Discussion

Our results and, in particular, the FPRs for specific thresholds are estimated using ImageNet. While ImageNet already contains more than 1M images, results might differ in different and, in particular, larger datasets. We randomly sample P images from ImageNet and, for each hashing algorithm, compute the distances between image hashes for up to 1M pairs. We find the distribution of distances between images to be roughly stable when the number of images varies for each algorithm. Similarly, we run the analysis on another dataset, Stanford Dogs [179], and show the distribution of distances to be similar to that of ImageNet and stable. This suggest that our FPRs can be safely extrapolated to larger and different datasets. Fig. 8.8 supports this analysis.

We have, throughout this work, assumed that the adversary knows the threshold and distance function used by the system to flag images. We believe this assumption to be reasonable as a) pairwise distances are fairly similar across datasets and stable, allowing an adversary to reasonably estimate acceptable thresholds and b) our attack works well, producing images very similar to the original one, even with very high thresholds. We also believe the assumption that the adversary knows the distance function used to be reasonable, as an adversary is likely to be able to infer the distance being used based on the output values, especially with the Euclidean and Hamming distance being the most commonly used.

While we report results on a database size of $N = 100$K, larger databases are often used in practice (e.g., the National Center for Missing and Exploited Children in the US reported a database size of 47.2M [5] and the Global Internet Forum for Counter Terrorism reported a database of size 250K [134]). Fig. 8.9 shows that the FPR of a system increases with the database size N. Even for the lowest threshold, the FPR for $N = 500$K reaches 0.51% for pHash continuous ($T = 0.6$), 0.17% ($T = 6$) for pHash, 6.44% ($T = 3$) for aHash, 0.16% ($T = 4$) for dHash and 0.11% ($T = 70$) for PDQ. This strongly suggest that the FPRs we report are a lower bound on the FPRs and therefore the privacy risk in practice.

We have so far considered our attack to be successful when the modified image is at a distance

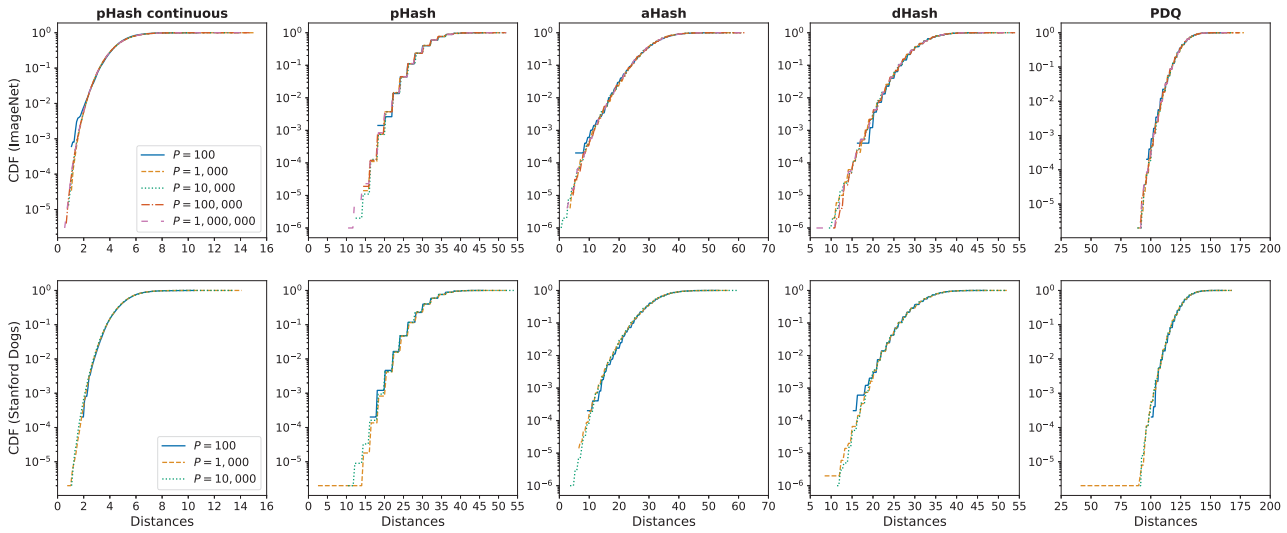


Figure 8.8: **Image distances across datasets.** Cumulative distribution function (CDF) for distances between the image hashes for different algorithms, number of images P and datasets. The CDFs are roughly stable for each algorithm for different number of images.

$> T$ from the original image. In practice, however, a modified image might be at a distance $> T$ from the original image but still be at a distance $\leq T$ from another image in the database and be (correctly, even if for the wrong reason) flagged. To evaluate the impact this has on the detection avoidance capability of our attack, we compute the False Negative Rate (FNR), the fraction of modified images incorrectly rejected by the system. Fig. 8.10 shows that the FNR stays extremely high for all perceptual hashing algorithms at reasonable thresholds, indicating that very few modified images get flagged for being at a distance $\leq T$ from another image in the database. As expected, FNR decreases (and FPR increases) when databases becomes large and a large threshold is used, leading to a significant fraction of the space being considered "close" to at least one illegal image. For instance, for a database size $N = 500$K and the highest threshold, 3.4% of modified images would still evade detection by pHash and 23.4% by PDQ.

Our black-box attack relies on a heuristic to minimise the perturbation while increasing the distance to the original image above a threshold T. Although we produce imperceptible perturbations for most of the attacked images, our results are only an upper estimate of the actual minimum perturbation. Prior work studying the robustness of machine learning classification models to adversarial perturbations showed that finding minimal perturbation is a hard problem [175] and even lower perturbations might be achieved in future work.

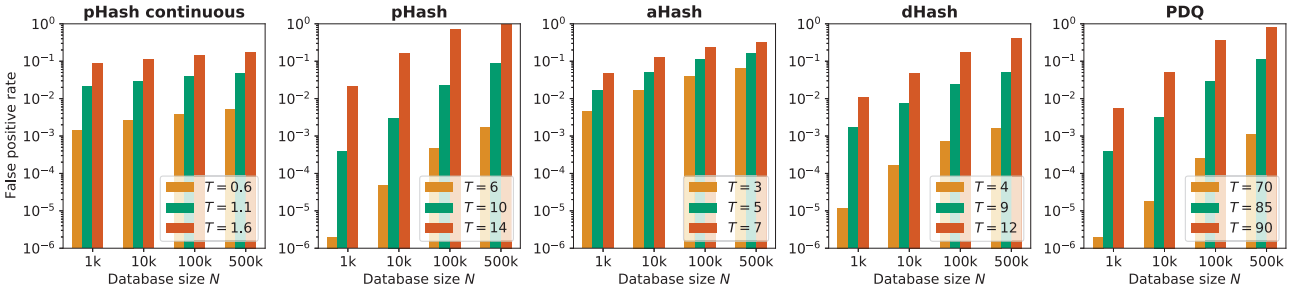


Figure 8.9: **False Positive Rate** (FPR) for each algorithm, threshold, and database size N. The number of images visually different from those in the database that are used to estimate the FPR is $M = 500\text{K}$.

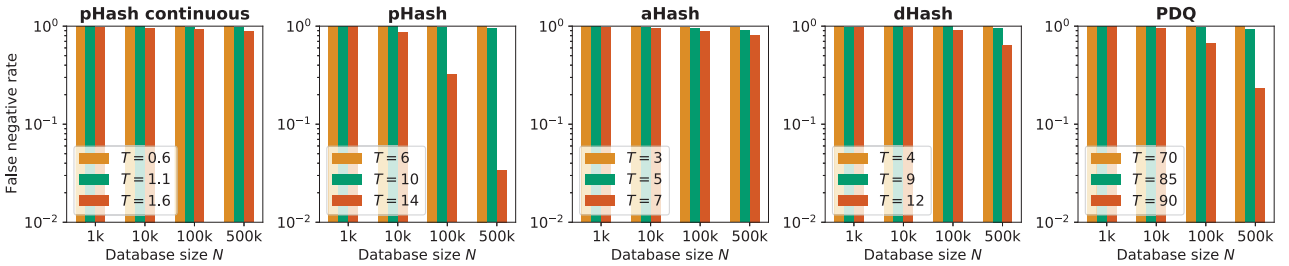


Figure 8.10: **False Negative Rate** (FNR) for each algorithm, threshold, and database size N. The number of attacked images used to estimate the FNR is $N' = 1000$.

We focus, in our white-box attack, on DCT-based perceptual hashing algorithms, exploiting the linearity of the DCT transform. Our attack is likely to be extendable to other linear image transforms, e.g. image scaling [350], opening the door to future attacks as well as potentially novel theoretical insights.

Countermeasures. We discuss several countermeasures that a detection system could implement to thwart our attack.

First, the system could use a larger threshold. We showed in Sec. 8.6 that the larger thresholds are not only ineffective in detecting adversarially modified images but also lead to a significant increase in false positives.

Second, one could use our attacks to generate hashes and add them to the database. We show in Sec. 8.6 that our attack is able to generate diverse perturbations. While the existence of diverse perturbations suggests that such countermeasures may be ineffective, we acknowledge two limitations of our current evaluation, which future work could address in order to very rigorously assess this claim. The first limitation is that we did not evaluate the impact on the

FPR of adding D perturbed images to the database for every image. This would increase the database size from N to $N \times D$. We have seen that when adding natural images to the database, the FPR increases. While we do not see any reason for the FPR not to increase similarly when diverse images are added, we believe that an evaluation of FPR for different thresholds T and values of D and N would allow to rigorously assess this claim. The second limitation is more subtle. In our evaluation we only demonstrated the existence of pairs of modified images whose hashes that are more than T away from one another and from the original image hash. In fact, demonstrating that D additions to the database are not enough to thwart our attack requires showing that a $(D+1)$-th perturbation which is at least T away from all the others can almost always be found. More precisely, more than one such perturbation would have to exist so as to make the attack non-deterministic. One way to tackle this requirement is to reformulate the optimisation problem so as to design perturbations one-by-one, with the additional requirement that every new perturbation should be T away from all the previous ones. In a small experiment, we implemented this approach with the DCT algorithm in the white-box setting. We found such perturbations to exist, although it takes more time to find them as D increases. We leave the design of a similar attack in the black-box setting for future work. To summarise, we conjecture that (1) for small values of D (e.g., ≤ 20) a relatively large number of diverse perturbations exist that are T away from one another and from the original image hash, making the addition of diverse perturbations ineffective and (2) for larger values of D, adding diverse perturbations to the database would increase the database size beyond acceptable values of the FPR.

Third, systems (such as Apple's protocol [1]) could flag users only after the number of matches exceed a predefined threshold k, rather than on a single match. Using a simple model, we show that such a measure is not a trivial countermeasure to our attack.

Our model consists of two types of users, each sending S images independently from one another. The first type, which we call non-offender, sends only non-illegal images. The second type, which we call offender, sends i illegal images that are adversarially modified using our attack and $S - i$ non-illegal images. The probability for the detection system to flag an image is FPR for non-illegal images and $1 - FNR$ for illegal images adversarially modified using our

attack.

Under this model, the number of images flagged for a non-offender follows a binomial distribution $S_1 \sim Bin(S, FPR)$. Similarly, the number of images flagged for an offender can be written as the sum of two binomial random variables $S_2 \sim Bin(i, 1-FNR) + Bin(S-i, FPR)$. The two follow the same distribution if $1 - FNR = FPR$, i.e., if the adversarial images are indistinguishable from natural images to the detection system.

This allows us to compute, for each type of user, the probability that $0 \leq k \leq S$ of their images are flagged:

$$P(S_1 = k) = \binom{S}{k} FPR^k (1 - FPR)^{S-k} \tag{8.24}$$

$$P(S_2 = k) = \sum_{i=0}^{min(i,k)} \binom{i}{l} (1 - FNR)^l FNR^{i-l} \times \binom{S-i}{k-l} FPR^{k-l} (1 - FPR)^{S-i-k+l} \tag{8.25}$$

We instantiate the model assuming that both offenders and non-offenders send $S = 1000$ images, and that offenders send exactly $i = 100$ illegal images (10%). For each type of user and each threshold, we compute the probability for the system to flag $1 \leq k \leq S$ images using the estimated FPR and FNR for a database size of 100K.

Fig 8.11 shows that offenders and non-offenders are similarly likely to have at least k images flagged. We obtain similar results for the other perceptual hashing algorithms (see Fig. 8.12). This can be attributed to the fact that the probability for a non-illegal image to get flagged (FPR) is comparable to the probability for an adversarially modified illegal image to be flagged (true positive rate or $TPR = 1 - FNR$). These results suggest that flagging a user with at least k matches is not a trivial countermeasure against our attack. We leave a deeper analysis of such a mechanism for future work.

Fourth, the detection system could apply additional image transformations before computing the perceptual hash of the image. Such modifications would however be part of the black-box and attacked by our algorithm. Simple transformations are thus unlikely to prevent our adversarial detection avoidance attack. Similarly, modifying the perceptual hashing algorithm

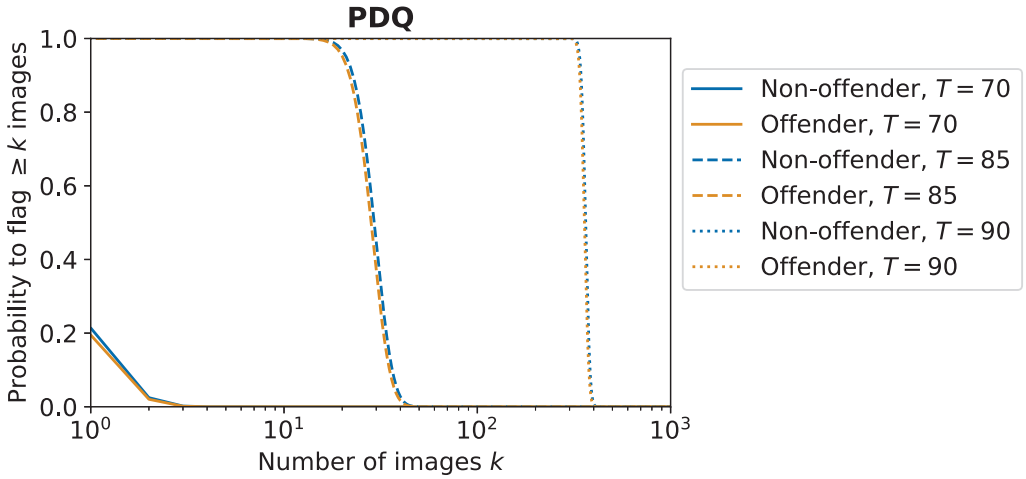


Figure 8.11: **Probability for PDQ to flag at least k images** shared by non-offenders (sharing 0 illegal images) and offenders (sharing 100 illegal images). Both offenders and non-offenders share a total of 1000 images.

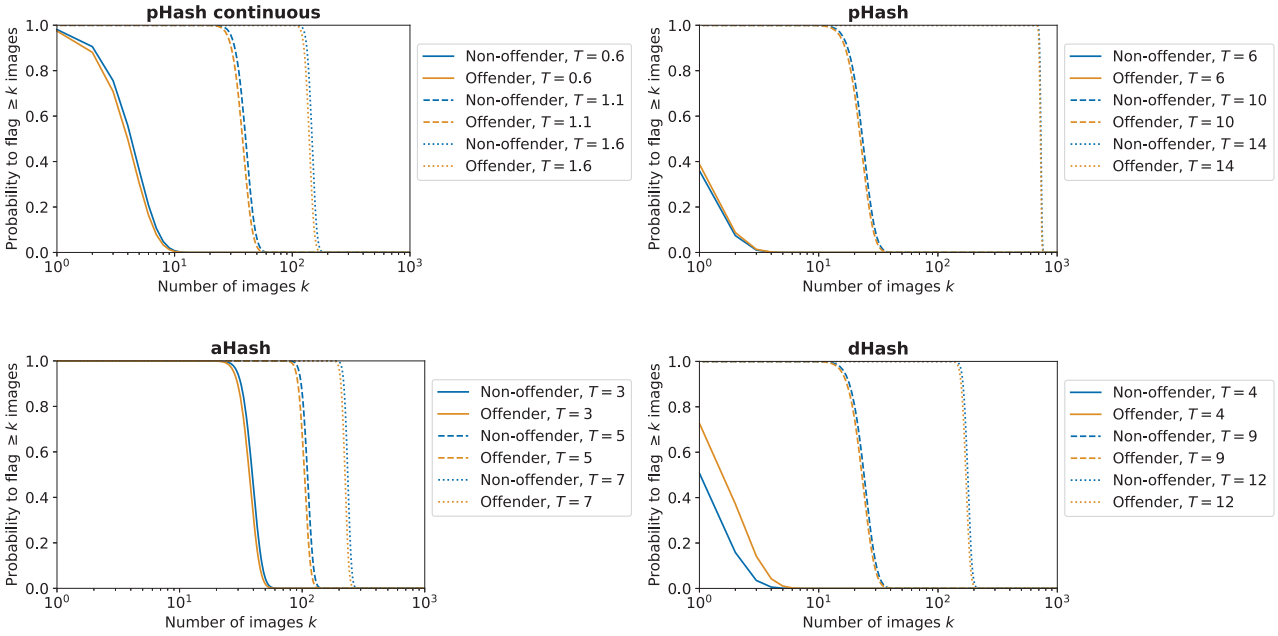


Figure 8.12: **Probability for other hashing algorithms to flag at least k images** shared by non-offenders (sharing 0 illegal images) and offenders (sharing 100 illegal images). Both offenders and non-offenders share a total of 1000 images.

for instance by using a secret set of DCT coefficients that is different from the set used by pHash or PDQ is unlikely to prevent our attack. These would furthermore make the algorithm less robust as the coefficients extracted by pHash and PDQ represent the most important features of the image.

Fifth, increasing the length of the hashes is likely to help better distinguish between different

images and reduce the false positive rate. This might allow the system to increase what are realistic thresholds and make our attack harder. We here use hashes of size 64 (and 256 for PDQ [178], see Sec. 8.5.2). Most perceptual hashing algorithms can be adapted to provide hashes of different sizes. For instance, pHash, aHash and dHash can all output hashes of size 256 [2]. While further analysis is required, we note here that a) our attack works well even for large thresholds and b) longer hashes encode more information about the image. This raises strong privacy concerns e.g. reversal attacks [204] and, in the extreme, defeats the purpose of perceptual hashing.

Finally, the system could try to detect if the image has been manipulated using our attack. Our analysis of the third countermeasure suggests that hashes of modified and natural images are indistinguishable to the CSS system, since the probability for a natural (non-illegal) image to be flagged is comparable to the probability of a modified illegal image to be flagged. However, an additional detection module could be added, e.g., by training a classifier to infer if the image was manipulated. Whether a classifier could accurately detect images modified using our attack is an open question. Assuming that it could, the possibility remains that new perturbations can be found which evade both the classifier and the CSS system. Research findings from adversarial machine learning (ML) indeed suggest that ad-hoc defenses can be thwarted at a later stage [55]. Future work could design and evaluate such a classifier as well as explore other defences. Furthermore, ML-based perceptual hashing algorithms could be designed so as to satisfy formal guarantees similar to certified robustness for ML models [74]. As for shallow perceptual hashing algorithms such as the ones studied in this work, an interesting venue is to develop new algorithms with formal robustness guarantees.

Implications. Our research shows that current perceptual hashing algorithms are not robust to black-box detection avoidance attacks and sheds strong doubt on the effectiveness of straightforward mitigations strategies. We believe perceptual hashing algorithms, which are designed to produce hashes that change gradually as the image changes, might be inherently vulnerable to attacks. Our results, combined with the concerns PH-CSS and in particular cryptographic-enhanced PH-CSS such as Apple's [17] raise on the "illegal" content being searched for, lead us

[2] https://github.com/thorn-oss/perception

to believe that even the best PH-CSS proposals today are not ready for deployment.

8.8 Related work

Robustness of perceptual hashing algorithms. Previous work have shown perceptual hashing algorithms to be robust to small standard image transformations like resizing, recoloring, watermarking, cropping, and blurring [362, 106]. They evaluate the performances of perceptual hashing algorithms against modifications in duplicate image detection setup, i.e. matching against a single image. They however focus on standard image transformations and do not consider adversaries with more sophisticated tools at their disposal.

Dolhansky and Ferrer [102] studied collision (false positive) attacks in perceptual hashing algorithms under white-box assumptions. More specifically, their algorithm modifies an image such that its hash is same as the hash of a given target image. They conclude that perceptual hashing algorithms should remain secret. In this work, we instead focus on the client-side use case proposed recently by researchers and policymakers [76, 216] where the attacker has a black-box access to the algorithm. Furthermore, our attack is a false negative attack, aiming at finding minimal diverse perturbations such that distance between the hashes of modified and input image is greater than a given threshold.

Adversarial attacks against ML models. Adversarial attacks have been widely studied against ML models [271, 298] under both white-box [322, 141, 189, 56] and black-box assumptions [252, 203]. Image classification models have been particularly found to be vulnerable to adversarial attacks [13] leading the image to be misclassified by the ML model. Our attack leverages previous work in adversarial ML including Natural Evolutionary Strategies (NES) (see Sec. 8.4.2).

Adversarial attacks on ML models in a black-box setting assume that the attacker has an access to query the model and get the output. A commonly observed approach is to train a substitute model to emulate the target model, and then attack the surrogate model [252, 301]. This approach implicitly assumes that the surrogate model has enough parameters to provide

a good approximation of the target model. This would imply that not all substitute models would work for all the target models and hence the attack might not always work.

Another approach is to iteratively perturb the image and query the model to update the perturbation in each iteration [144]. These methods often use gradient estimation to estimate the perturbation update in each iteration. Natural evolutionary strategies (NES) for gradient estimation is one of the state-of-the-art methods in black-box adversarial ML [65, 161, 218]. Our black-box attack also uses NES-based strategy for gradient estimation and perturbation update. Lastly, we note that generating multiple perturbations for the same image for diversity in the output space is not a requirement in the adversarial ML setup, while it is an important requirement for adversarial attacks against PH-CSS.

8.9 Conclusion

In this work, we introduced the first framework to evaluate the robustness of perceptual hashing-based client-side scanning against adversarial attacks. We proposed a general black-box attack and showed that $> 99.9\%$ of images can be successfully modified while preserving the image content. We also show our attack to generate diverse perturbations which suggests that straightforward mitigation strategies such as expanding the database with modified images would be ineffective. We finally propose two white-box attacks, providing a theoretical basis for attacks. Taken together, our results shed strong doubt on the robustness to adversarial black-box attacks of perceptual hashing-based client-side scanning as currently proposed. The detection thresholds necessary to make the attack harder are likely to be very large, probably requiring more than one billion images to be wrongly flagged daily, raising strong privacy concerns.



Figure 8.13: **First example of an original image with modifications obtained using our black-box attack for different perceptual hashing algorithms and thresholds.** We show the $\mathbb{L}_2$ perturbation per pixel. All the modified images are qualitatively very similar to the original image.



Figure 8.14: **Second example of an original image with modifications obtained using our black-box attack for different perceptual hashing algorithms and thresholds.** We show the $\mathbb{L}_2$ perturbation per pixel. All the modified images are qualitatively very similar to the original image.

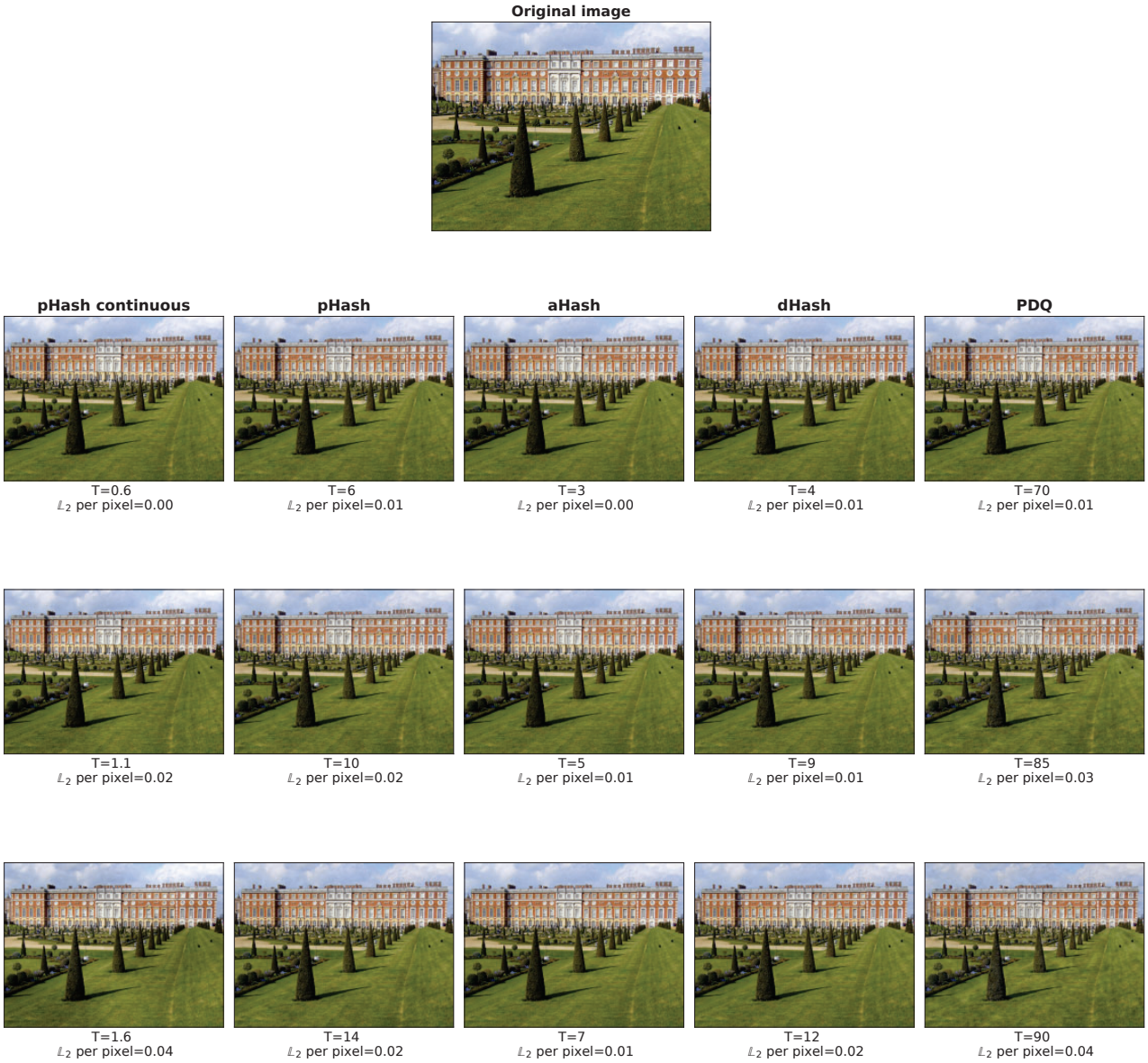


Figure 8.15: **Third example of an original image with modifications obtained using our black-box attack for different perceptual hashing algorithms and thresholds.** We show the $\mathbb{L}_2$ perturbation per pixel. All the modified images are qualitatively very similar to the original image.

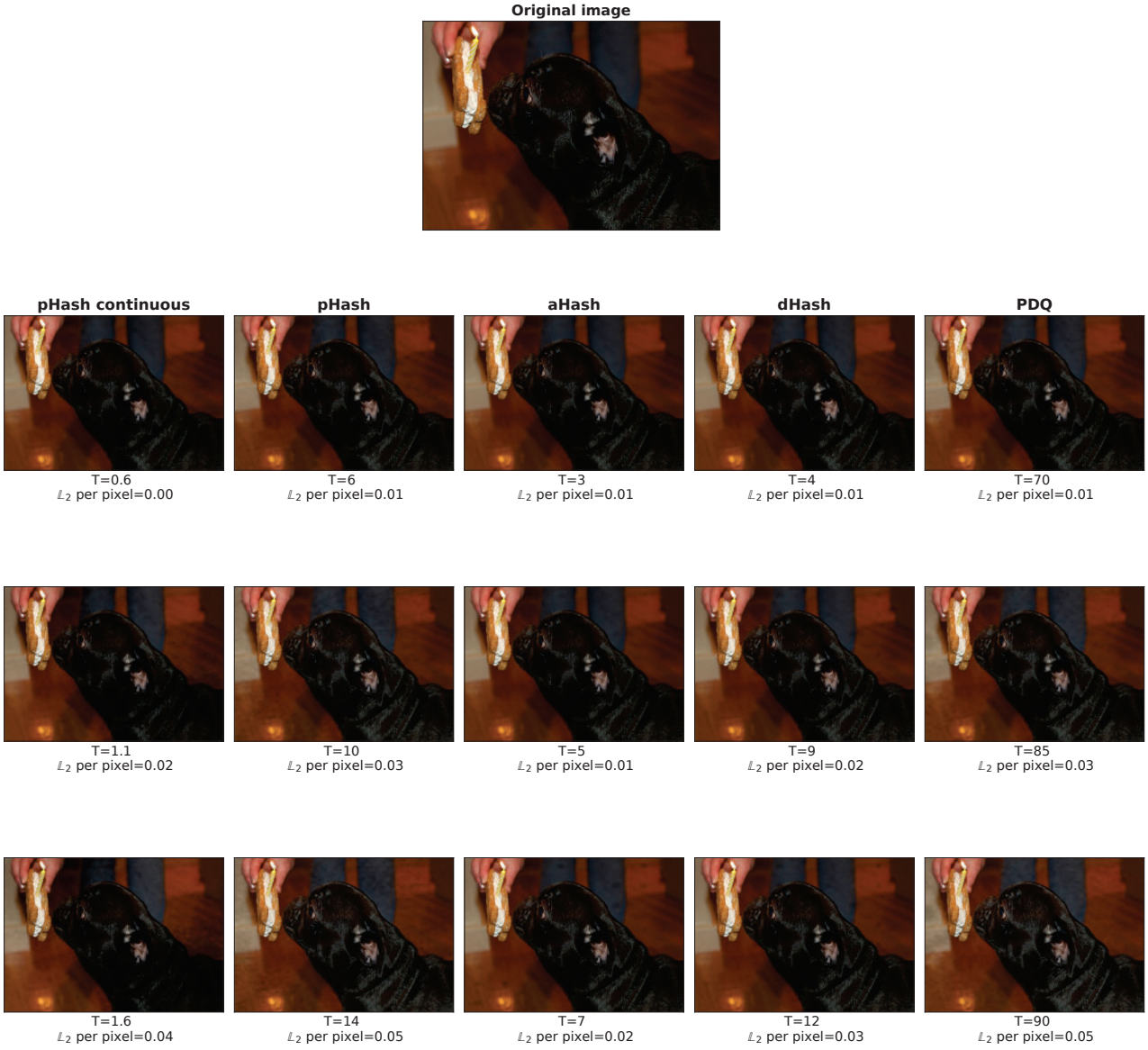


Figure 8.16: **Fourth example of an original image with modifications obtained using our black-box attack for different perceptual hashing algorithms and thresholds.** We show the $\mathbb{L}_2$ perturbation per pixel. All the modified images are qualitatively very similar to the original image.



Figure 8.17: **Fifth example of an original image with modifications obtained using our black-box attack for different perceptual hashing algorithms and thresholds.** We show the $\mathbb{L}_2$ perturbation per pixel. All the modified images are qualitatively very similar to the original image.

Chapter 9

Conclusion

9.1 Summary of thesis achievements

This thesis contributes to the field of privacy by developing new threat models and attack methods against four different technologies to process or share data under five different contexts. We summarise our contributions and future directions below.

In Chapter 3, we demonstrate how people's interaction behaviour is stable over multiple weeks and can be used to identify individuals in pseudonymised and frequently re-pseudonymised datasets. We propose a graph neural network embedding model that extracts a profile of an individual which can be used to identify them across different time periods. In a mobile phone metadata dataset of more than 40K people, it correctly identifies 52% of individuals based on their 2-hop interaction graph. We further show that the profiles learned by our method are stable over time and that 24% of people are still identifiable after 20 weeks. Applying our attack to Bluetooth close-proximity networks, we show that even 1-hop interaction graphs are enough to identify people more than 26% of the time. Our results provide strong evidence that disconnected and even re-pseudonymised interaction data can be linked together making them personal data under the EU's GDPR.

This work could serve as the basis for a more extensive study of the robustness of re-pseudonymi-

sation to deep learning profiling attacks, to (1) explore the impact of reducing the frequency of re-pseudonymisation from a week to a few days on the privacy and utility of the data and (2) apply the attack to other dynamic data domains, such as smart meter and financial transaction data.

In Chapter 4, we develop QuerySnout, the first approach to automatically discover vulnerabilities in query-based systems (QBS). QuerySnout takes as input a target record and the QBS as a black box, analyses its behaviour on one or more datasets, and outputs a multiset of queries together with a rule to combine answers to them in order to reveal the sensitive attribute of the target record. QuerySnout uses evolutionary search techniques based on a mutation operator tailored to the problem to find a multiset of queries susceptible to lead to an attack, and a machine learning classifier to infer the sensitive attribute from answers to the queries selected. We show QuerySnout to find attacks against two deterministic real-world mechanisms, Diffix and TableBuilder, and a non-deterministic system implementing query set size restriction and Gaussian noise addition. Across systems and datasets, the attacks found equate and often outperform previous known manual attacks. Finally, we show how QuerySnout can be extended to QBSes that require a budget and show it to approximately match the optimal accuracy against systems implementing the Laplace mechanism for large values of ε. We believe that automated attack discovery procedures such as QuerySnout will help detect issues before systems are deployed, helping to patch the systems and mitigate risks, and ultimately help design query-based systems offering a high level of protection in practice.

Further work is needed to extend QuerySnout to other settings. In particular, we highlight two important research directions: (1) extending the query syntax and (2) relaxing the assumptions on the adversary's knowledge about the private dataset, by assuming the adversary only knows a few summary statistics (which can also be retrieved from the QBS) and using them to generate synthetic shadow datasets.

In Chapter 6, we study a new type of leakage in machine learning (ML) models: the leakage of correlations between input variables of a tabular training dataset. While machine learning models are often trained on sensitive and proprietary datasets, the fundamental question of

what information a model "remembers" about its training dataset and under what conditions the leakage of such information is possible is not well understood. We develop the first correlation inference attack against ML models, proposing two different approaches: (1) a model-less attack that explores the spherical parameterisation of correlation matrices to make an informed guess based on the correlations between the input variables and the target variable alone, and (2) a model-based attack extending the shadow modelling technique to synthetic datasets generated using the Gaussian copula model that allows to control for different values of the correlations. We perform an extensive evaluation of our attacks, showing that models leak correlations of their training dataset with high accuracy, in both synthetic and real-world datasets. We also show that mitigations such as limiting the number of queries that can be made to the model, their precision, or training the models with differential privacy guarantees do not help. Finally, we show how correlations leaked from the model using our attack can be used as building blocks for attribute inference attacks.

Our work provides the first step towards a more comprehensive evaluation of what ML models learn. In particular, it could be extended (1) to perform the correlation inference attack in an end-to-end fashion, by first extracting the correlations constraints (between the input variable and the target variables) from the model before running the attack and (2) to infer more detailed information about the training distribution, such as k-way marginals.

In Chapter 7, we propose a new white-box membership inference attack (MIA) against ML models using re-aligned shadow models. We first study the causes of shadow model misalignment, where the different models learn similar features in each layer but the features are located in different positions, showing the use of a different model initialisation to be the main cause. We then evaluate several techniques to re-align the layers of a shadow model to those of the target model, showing them to significantly reduce the measured misalignment. Finally, we show misalignment to affect the performance of white-box membership inference attacks and evaluate the ability of re-aligned shadow models to enhance their performance on three datasets. Our results highlight the importance of addressing neural network misalignment when performing white-box shadow modelling attacks.

Future work could (1) evaluate the performance of re-alignment techniques on property inference attacks, comparing their performance with set-based classifiers and (2) develop better re-alignment techniques that further reduce misalignment.

In Chapter 8, we evaluate the robustness of perceptual hashing-based client-side scanning (PH-CSS) to adversarial detection avoidance attacks. As end-to-end encryption (E2EE) has been argued to facilitate the sharing of illegal content such as child sexual abuse material (CSAM), PH-CSS has been recently proposed by tech companies, governments and researchers as a solution to detect CSAM in E2EE communications. PH-CSS solutions detect CSAM content directly the user devices before encryption, by computing a fingerprint of every image shared by a user using a perceptual hashing algorithm, and comparing it against a database of fingerprints of known CSAM images. Flagged images would be shared raw (unencrypted) with relevant authorities for a manual review. Significant privacy concerns have been raised, including about the system's potential to falsely flag non-CSAM images, as sharing such images defeats the purpose of E2EE and could lead to privacy violations. A key question is thus whether the privacy and security risks of PH-CSS, such as the risk of false positives, can be balanced by effective detection. Our contribution is the first framework to evaluate the robustness of PH-CSS to detection avoidance attacks. We propose a general black-box attack and showed that $> 99.9\%$ of images can be successfully modified while preserving the image content. We also show our attack to generate diverse perturbations, shedding doubt on the effectiveness of straightforward mitigation strategies such as expanding the database with modified images. We finally propose two white-box attacks, providing a theoretical basis for attacks. Our results shed strong doubt on the robustness to adversarial black-box detection avoidance attacks of currently proposed PH-CSS solutions. The detection thresholds necessary to make the attack harder are likely to be very large, probably requiring more than one billion images to be wrongly flagged daily, raising strong privacy concerns.

More research is needed to examine the privacy and security properties of PH-CSS solutions. This line of research is particularly timely considering current government proposals of legislation in the UK [121] and the EU [6] that would provide law enforcement agencies with powers to mandate their deployment on people's devices.

9.2 Perspectives

In this thesis, we have taken an *adversarial empirical approach* to privacy. In each situation we have studied, we aimed to model adversaries (their identity, goal, and auxiliary knowledge) and used state-of-the-art machine learning (ML) techniques to design powerful adversaries. We believe this adversarial empirical approach presents several advantages (while we here restrict this discussion to the privacy case, similar arguments can be made for PH-CSS).

First, it complements the other dominant approach of defining *a priori* a set of privacy guarantees the data release should satisfy, such as differential privacy (DP) [107], by allowing to empirically test if such guarantees hold in practice. Indeed, while DP ensures protection against a wide range of attacks, implementations may contain errors that invalidate its formal guarantees. Researchers have indeed shown that implementations of DP are vulnerable to floating-point [220, 170, 57] and timing attacks [170].

Second, the adversarial empirical approach is particularly well-suited in situations where analysing a system "on paper" is very difficult or impossible. One example is ad-hoc anonymisation in QBSs: real-world systems tend to be very complex, allowing for a very large syntax set and adding different levels of noise to different queries. Another example is ML models, where little is known about what exactly a model remembers about its training dataset D: inference attacks fill this gap by measuring whether a model leaks a certain secret about the dataset and under what conditions.

Finally, the adversarial empirical approach is context-dependent, i.e., it can allow the analysis of privacy guarantees "in practice", whereas DP provides an upper bound by protecting even against the "worst-case" adversary. This strength of the adversarial approach has been recognised recently in the topic of auditing differentially private machine learning (DP ML). A challenge of DP ML is that the values of ϵ necessary to achieve meaningful privacy guarantees greatly affect the utility of the model. The privacy budget ϵ limits the worst-case leakage by an adversary having access to the complete dataset but one record and to all intermediate models obtained during training [8]. However, realistic adversaries that practitioners are worried about

are much weaker, as they typically have uncertainty about the dataset and only access to the final model. It is currently very difficult to analyse theoretically the privacy afforded against such adversaries. The adversarial approach can help fill this gap by empirically lower bounding the value of ϵ [163, 230]. A similar argument applies to other aggregate data releases such as synthetic data [158].

Further work is however needed to extend the applicability of the adversarial approach for the analysis of privacy-enhancing technologies. We highlight two limitations.

The first limitation is that attacks, including ours, are often implemented in artificial settings ("lab" conditions). For instance, the adversary access to auxiliary data similar in distribution to the private dataset (chapters 4 and 7) is a particularly stringent assumption that is unlikely to be satisfied by real-world adversaries. Future work could, first, investigate the use of synthetic datasets sampled under constraints given by the adversary knowledge, e.g., query answers in the case of QBS, to replace the auxiliary data assumption that is so common in the literature. This would enable the evaluation of the risks posed by weaker (more realistic) adversaries. In chapter 6, we show how this can be done in the case of correlations. Second, more research is needed to study the risk of privacy and property inference attacks "in the wild", e.g., on actual models deployed on devices, without additional assumptions on adversary knowledge to data or training algorithm. This could help bridge the gap between the high risk values reported in the literature and what an adversary is likely to achieve in the real world.

The second limitation is that attacks only bring a lower bound on the risk, and in theory one would have to try all of them to correctly estimate the risk, which is infeasible. In addition to this inherent limitation, automating attacks through machine learning and other automated techniques (e.g., evolutionary search) can be computationally very expensive. While it may be possible to show that an optimal attack exists, in many cases this optimal attack is computationally expensive to perform. Take for instance a binary inference attack, where the adversary wants to learn the answer (yes/no) to a question about a dataset, such as whether Alice's record was used to train a model. The natural way to answer this question is via the hypothesis testing framework, by distinguishing between two worlds: the world of data releases (ML

model, QBS, etc.) derived from datasets where the answer is yes and the world of data releases derived from datasets where the answer is no [52]. Under the hypothesis testing framework, it is necessary to instantiate the data release, e.g., train a model, on many datasets, half of which have the answers yes (and the other half no). This approach is known as shadow modelling and is extremely resource demanding. How to replace current shadow modelling pipelines with simpler, but equally effective approaches is an open problem. Solving this problem could help reduce the cost of attacks and enable a rigorous analysis of the privacy of larger systems, such as large language models.

Using data for good while respecting the privacy of individuals is one of the great challenges of the digital age. Research so far has shown that it is often very difficult to reconcile utility and privacy. This thesis provides evidence that the adversarial empirical approach is a promising way to evaluate the privacy and security of data processing technologies, especially when combined with advanced machine learning and automation.

Bibliography

[1] CSAM Detection - Technical Summary 2021. https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf.

[2] General data protection regulation. https://gdpr-info.eu/, 2016.

[3] California state legislature, california consumer privacy act of 2018. https://www.consumerprivacyact.com/section-1798-140-definitions/, 2018.

[4] Guidelines 04/2020 on the use of location data and contact tracing tools in the context of the covid-19 outbreak. https://edpb.europa.eu/sites/edpb/files/files/file1/edpb_guidelines_20200420_contact_tracing_covid_with_annex_en.pdf, 2020.

[5] The internet investigation report march 2020. https://www.iicsa.org.uk/document/internet-investigation-report-march-2020, 2020. Accessed on June 8, 2021.

[6] *Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL laying down rules to prevent and combat child sexual abuse.* 2022.

[7] M. Abadi. Tensorflow: learning functions at scale. In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*, pages 1–1, 2016.

[8] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[9] H. Abelson, R. Anderson, S. M. Bellovin, J. Benaloh, M. Blaze, J. Callas, W. Diffie,

S. Landau, P. G. Neumann, R. L. Rivest, et al. Bugs in our pockets: The risks of client-side scanning. *arXiv preprint arXiv:2110.07450*, 2021.

[10] J. P. Achara, G. Acs, and C. Castelluccia. On the unicity of smartphone applications. In *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society*, pages 27–36, 2015.

[11] N. Aharony, W. Pan, C. Ip, I. Khayal, and A. Pentland. Social fmri: Investigating and shaping social mechanisms in the real world. *Pervasive and mobile computing*, 7(6):643–659, 2011.

[12] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

[13] N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430, 2018.

[14] A. Aktay, S. Bavadekar, G. Cossoul, J. Davis, D. Desfontaines, A. Fabrikant, E. Gabrilovich, K. Gadepalli, B. Gipson, M. Guevara, et al. Google covid-19 community mobility reports: anonymization process description (version 1.1). *arXiv preprint arXiv:2004.04145*, 2020.

[15] Y. Altshuler, N. Aharony, M. Fire, Y. Elovici, and A. Pentland. Incremental learning with accuracy prediction of social and individual properties from mobile-phone data. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 969–974. IEEE, 2012.

[16] M. Alzantot, Y. Sharma, S. Chakraborty, H. Zhang, C.-J. Hsieh, and M. B. Srivastava. Genattack: Practical black-box attacks with gradient-free optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1111–1119, 2019.

[17] Apple. Expanded protections for children. `https://www.apple.com/child-safety/`. Accessed on Sep 18, 2021.

[18] Apple and Google. Privacy-preserving contact tracing. `https://www.apple.com/covid19/contacttracing`, 2020.

[19] Article 29 Data Protection Working Party. Opinion 05/2014 on anonymisation techniques. `https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2014/wp216_en.pdf`, 2014.

[20] H. J. Asghar and D. Kaafar. Averaging attacks on bounded noise-based disclosure control algorithms. *Proceedings on Privacy Enhancing Technologies*, 2:358–378, 2020.

[21] S. Ashmore and M. Gashler. A method for finding similarity between multi-layer perceptrons by forward bipartite alignment. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2015.

[22] U. G. Assembly et al. Universal declaration of human rights. *UN General Assembly*, 302(2):14–25, 1948.

[23] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks*, 10(3):137–150, 2015.

[24] S. Aydore, W. Brown, M. Kearns, K. Kenthapadi, L. Melis, A. Roth, and A. Siva. Differentially private query release through adaptive projection. *arXiv preprint arXiv:2103.06641*, 2021.

[25] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x? anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*, pages 181–190, 2007.

[26] E. Bagdasaryan, O. Poursaeed, and V. Shmatikov. Differential privacy has disparate impact on model accuracy. *Advances in neural information processing systems*, 32, 2019.

[27] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[28] R. Balestriero, M. Ibrahim, V. Sobal, A. Morcos, S. Shekhar, T. Goldstein, F. Bordes, A. Bardes, G. Mialon, Y. Tian, et al. A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*, 2023.

[29] B. Balle, G. Cherubin, and J. Hayes. Reconstructing training data with informed adversaries. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1556–1556. IEEE Computer Society, 2022.

[30] M. Barbaro, T. Zeller, and S. Hansell. A face is exposed for aol searcher no. 4417749. *New York Times*, 9(2008):8, 2006.

[31] D. Barth-Jones, K. E. Emam, J. Bambauer, A. Cavoukian, and B. Malin. Assessing data intrusion threats. *Science*, 348(6231):194–195, 2015.

[32] F. Bauer. Announcing the first ever bug bounty program for a privacy protection solution, october 2017.

[33] F. Bauer. The world's only anonymization bug bounty – round two!, May 2020.

[34] L. Bengtsson, J. Gaudart, X. Lu, S. Moore, E. Wetter, K. Sallah, S. Rebaudet, and R. Piarroux. Using mobile phone data to predict the spatial spread of cholera. *Scientific reports*, 5(1):8923, 2015.

[35] B. Bichsel, S. Steffen, I. Bogunovic, and M. Vechev. Dp-sniper: black-box discovery of differential privacy violations using classifiers. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 391–409. IEEE, 2021.

[36] W. blog. Connecting one billion users every day. `https://blog.whatsapp.com/connecting-one-billion-users-every-day`, 2017. Accessed on June 6, 2021.

[37] W. Blog. Two billion users – connecting the world privately. `https://blog.whatsapp.com/two-billion-users-connecting-the-world-privately/`, 2020. Accessed on June 6, 2021.

[38] V. Blondel, G. Krings, and I. Thomas. Regions and borders of mobile telephony in belgium and in the brussels metropolitan zone. *Brussels Studies. La revue scientifique*

pour les recherches sur Bruxelles/Het wetenschappelijk tijdschrift voor onderzoek over Brussel/The Journal of Research on Brussels, 2010.

[39] V. D. Blondel, M. Esch, C. Chan, F. Clérot, P. Deville, E. Huens, F. Morlot, Z. Smoreda, and C. Ziemlicki. Data for development: the d4d challenge on mobile phone data. *arXiv preprint arXiv:1210.0137*, 2012.

[40] J. Blumenstock, G. Cadamuro, and R. On. Predicting poverty and wealth from mobile phone metadata. *Science*, 350(6264):1073–1076, 2015.

[41] F. Boenisch, R. Munz, M. Tiepelt, S. Hanisch, C. Kuhn, and P. Francis. Side-channel attacks on query-based data anonymization. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1254–1265, 2021.

[42] R. M. Bond, C. J. Fariss, J. J. Jones, A. D. Kramer, C. Marlow, J. E. Settle, and J. H. Fowler. A 61-million-person experiment in social influence and political mobilization. *Nature*, 489(7415):295–298, 2012.

[43] M. Boorstein, M. Iati, and A. Shin. Top u.s. catholic church official resigns after cellphone data used to track him on grindr and to gay bars. `https://www.washingtonpost.com/religion/2021/07/20/bishop-misconduct-resign-burrill/`, 2021.

[44] A. Bovet and H. A. Makse. Influence of fake news in twitter during the 2016 us presidential election. *Nature communications*, 10(1):7, 2019.

[45] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[46] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[47] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[48] BT. BT.com privacy policy. https://img01.products.bt.co.uk/content/dam/bt/storefront/pdfs/BT.comcurrentprivacypolicy_18052018.pdf, 2018. accessed on 30/03/2020.

[49] C. Cadwalladr and E. Graham-Harrison. Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach. *The Guardian*, 2018.

[50] J. Callas. Thoughts on mitigating abuse in an end-to-end world. page 16, Jan 2020.

[51] R. Canetti, A. Trachtenberg, and M. Varia. Anonymous collocation discovery: Harnessing privacy to tame the coronavirus. *arXiv preprint arXiv:2003.13670*, 2020.

[52] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1897–1914. IEEE, 2022.

[53] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, volume 267, 2019.

[54] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. B. Brown, D. Song, U. Erlingsson, et al. Extracting training data from large language models. In *USENIX Security Symposium*, volume 6, 2021.

[55] N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 3–14, 2017.

[56] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[57] S. Casacuberta, M. Shoemate, S. Vadhan, and C. Wagaman. Widespread underestimation of sensitivity in differentially private libraries and how to fix it. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 471–484, 2022.

[58] C. Castelluccia, N. Bielova, A. Boutet, M. Cunche, C. Lauradoux, D. L. Métayer, and V. Roca. Desire: A third way for a european exposure notification system leveraging the best of centralized and decentralized systems. *arXiv preprint arXiv:2008.01621*, 2020.

[59] B. P. Chamberlain, C. Humby, and M. P. Deisenroth. Probabilistic inference of twitter users' age based on what they follow. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings, Part III 10*, pages 191–203. Springer, 2017.

[60] K. Chatzikokolakis, G. Cherubin, C. Palamidessi, and C. Troncoso. Bayes security: A not so average metric. In *2023 2023 IEEE 36th Computer Security Foundations Symposium (CSF)(CSF)*, pages 159–177. IEEE Computer Society, 2022.

[61] K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. *Advances in neural information processing systems*, 21, 2008.

[62] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

[63] B. Chen, M. Wen, Y. Shi, D. Lin, G. K. Rajbahadur, and Z. M. Jiang. Towards training reproducible deep learning models. In *Proceedings of the 44th International Conference on Software Engineering*, pages 2202–2214, 2022.

[64] J. Cheng, A. W.-c. Fu, and J. Liu. K-isomorphism: privacy preserving network publication against structural attacks. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 459–470, 2010.

[65] M. Cheng, T. Le, P.-Y. Chen, H. Zhang, J. Yi, and C.-J. Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *International Conference on Learning Representations*, 2018.

[66] U. Cherubini, E. Luciano, and W. Vecchiato. *Copula methods in finance*. John Wiley & Sons, 2004.

[67] J. Chipperfield, D. Gow, and B. Loong. The australian bureau of statistics and releasing frequency tables via a remote server. *Statistical Journal of the IAOS*, 32(1):53–64, 2016.

[68] C. A. Choquette-Choo, F. Tramer, N. Carlini, and N. Papernot. Label-only membership inference attacks. In *International conference on machine learning*, pages 1964–1974. PMLR, 2021.

[69] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pages 192–204. PMLR, 2015.

[70] S. Cleemput, M. A. Mustafa, E. Marin, and B. Preneel. De-pseudonymization of smart metering data: Analysis and countermeasures. In *2018 Global Internet of Things Summit (GIoTS)*, pages 1–6. IEEE, 2018.

[71] A. Cohen. Attacks on deidentification's defenses. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1469–1486, 2022.

[72] A. Cohen, S. Nikolov, S. Zachary, and J. Ullman. Reconstruction attacks in practice. `https://differentialprivacy.org/diffix-attack/`, October 2020. Accessed: 2021-6-25.

[73] A. Cohen and K. Nissim. Linear Program Reconstruction in Practice. *TPDP 2018*, Oct. 2018.

[74] J. Cohen, E. Rosenfeld, and Z. Kolter. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*, pages 1310–1320. PMLR, 2019.

[75] E. Commission. On artificial intelligence-a european approach to excellence and trust, 2020. `https://ec.europa.eu/info/sites/default/files/commission-white-paper-artificial-intelligence-feb2020_en.pdf`.

[76] E. Commission. Technical solutions to detect child sexual abuse in end-to-end encryption communications, 2020.

[77] G. Cormode. Personal privacy vs population privacy: learning to attack anonymization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1253–1261, 2011.

[78] B. Coskun and B. Sankur. Robust video hash extraction. In *2004 12th European Signal Processing Conference*, pages 2295–2298. IEEE, 2004.

[79] J. Cox, L. Franceschi-Bicchieral, and S. Cole. Apple defends its anti-child abuse imagery tech after claims of 'hash collisions'.

[80] A.-M. Creţu, F. Guépin, and Y.-A. de Montjoye. Correlation inference attacks against machine learning models. *arXiv preprint arXiv:2112.08806*, 2021.

[81] A.-M. Creţu, F. Monti, S. Marrone, X. Dong, M. Bronstein, and Y.-A. de Montjoye. Interaction data are identifiable even across long periods of time. *Nature Communications*, 13(1):313, 2022.

[82] R. Cummings, G. Kaptchuk, and E. M. Redmiles. " i need a better description": An investigation into user expectations for differential privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3037–3052, 2021.

[83] H. J. Curtis and B. Goldacre. Openprescribing: normalised data and software tool to research trends in english nhs primary care prescribing 1998–2016. *BMJ open*, 8(2):e019921, 2018.

[84] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song. Adversarial attack on graph structured data. In *International conference on machine learning*, pages 1115–1124. PMLR, 2018.

[85] T. Dalenius. Towards a methodology for statistical disclosure control. 1977.

[86] A. Davis and G. Rosen. Open-sourcing photo- and video-matching technology to make the internet safer, 2019. `https://about.fb.com/news/2019/08/open-source-photo-video-matching/`.

[87] S. De Capitani Di Vimercati, S. Foresti, G. Livraga, and P. Samarati. Data privacy: Definitions and techniques. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 20(06):793–817, 2012.

[88] K. A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD Thesis. University of Michigan, 1975.

[89] Y.-A. de Montjoye, S. Gambs, V. Blondel, G. Canright, N. De Cordes, S. Deletaille, K. Engø-Monsen, M. Garcia-Herranz, J. Kendall, C. Kerry, et al. On the privacy-conscientious use of mobile phone data. *Scientific data*, 5(1):1–6, 2018.

[90] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3(1):1–5, 2013.

[91] Y.-A. de Montjoye, J. Quoidbach, F. Robic, and A. Pentland. Predicting personality using novel mobile phone-based metrics. In *Social Computing, Behavioral-Cultural Modeling and Prediction: 6th International Conference, SBP 2013, Washington, DC, USA, April 2-5, 2013. Proceedings 6*, pages 48–55. Springer, 2013.

[92] Y.-A. de Montjoye, L. Radaelli, V. K. Singh, and A. S. Pentland. Unique in the shopping mall: On the reidentifiability of credit card metadata. *Science*, 347(6221):536–539, 2015.

[93] Y.-A. de Montjoye, L. Rocher, and A. S. Pentland. bandicoot: A python toolbox for mobile phone metadata. *The Journal of Machine Learning Research*, 17(1):6100–6104, 2016.

[94] Y.-A. de Montjoye, Z. Smoreda, R. Trinquart, C. Ziemlicki, and V. D. Blondel. D4d-senegal: the second mobile phone data for development challenge. *arXiv preprint arXiv:1407.4885*, 2014.

[95] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[96] D. E. Denning. Secure statistical databases with random sample queries. *ACM Transactions on Database Systems (TODS)*, 5(3):291–315, 1980.

[97] D. E. Denning and P. J. Denning. The tracker: A threat to statistical database security. *ACM Transactions on Database Systems (TODS)*, 4(1):76–96, 1979.

[98] D. Desfontaines and B. Pejó. Sok: Differential privacies. *Proceedings on Privacy Enhancing Technologies*, 2:288–313, 2020.

[99] C. Deußer, S. Passmann, and T. Strufe. Browsing unicity: On the limits of anonymizing web tracking data. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 777–790. IEEE, 2020.

[100] T. Dietterich. UCI Repository of machine learning databases: Musk (Version 2) Data Set, 1998. `https://archive.ics.uci.edu/ml/datasets/Musk+(Version+2)`.

[101] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 202–210, 2003.

[102] B. Dolhansky and C. C. Ferrer. Adversarial collision attacks on image hashing functions. *arXiv:2011.09473*, 2020.

[103] A. V. Dorogush, V. Ershov, and A. Gulin. Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018.

[104] M. Douze, G. Tolias, E. Pizzi, Z. Papakipos, L. Chanussot, F. Radenovic, T. Jenicek, M. Maximov, L. Leal-Taixé, I. Elezi, et al. The 2021 image similarity dataset and challenge. *arXiv preprint arXiv:2106.09672*, 2021.

[105] S. B. Driss, M. Soua, R. Kachouri, and M. Akil. A comparison study between mlp and convolutional neural network models for character recognition. In *Real-Time Image and Video Processing 2017*, volume 10223, pages 32–42. SPIE, 2017.

[106] A. Drmic, M. Silic, G. Delac, K. Vladimir, and A. S. Kurdija. Evaluating robustness of perceptual image hashing algorithms. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2017.

[107] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[108] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[109] C. Dwork, A. Smith, T. Steinke, J. Ullman, and S. Vadhan. Robust traceability from trace amounts. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 650–669. IEEE, 2015.

[110] C. Dwork and S. Yekhanin. New efficient attacks on statistical disclosure control mechanisms. In *Annual International Cryptology Conference*, pages 469–480. Springer, 2008.

[111] N. Eagle and A. Pentland. Reality mining: sensing complex social systems. *Personal and ubiquitous computing*, 10:255–268, 2006.

[112] Electronic Frontier Foundation. Why metadata matters. `https://ssd.eff.org/en/module/why-metadata-matters`, Mar 2019.

[113] R. Entezari, H. Sedghi, O. Saukh, and B. Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*, 2021.

[114] European Data Protection Board. Guidelines 4/2019 on Article 25 - Data Protection by Design and by Default. `https://edpb.europa.eu/sites/edpb/files/consultation/edpb_guidelines_201904_dataprotection_by_design_and_by_default.pdf`, 2020.

[115] C. Fabiana, M. Garetto, and E. Leonardi. De-anonymizing scale-free social networks by percolation graph matching. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 1571–1579. IEEE, 2015.

[116] A. Farzanehfar, F. Houssiau, and Y.-A. de Montjoye. The risk of re-identification remains high even in country-scale location datasets. *Patterns*, 2(3), 2021.

[117] B. Felbo, P. Sundsøy, A. S. Pentland, S. Lehmann, and Y.-A. de Montjoye. Modeling the temporal nature of human behavior for demographics prediction. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings, Part III 10*, pages 140–152. Springer, 2017.

[118] V. Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

[119] W. Feller. Probability theory and its applications, vol. 1 new york, 1968.

[120] D. for DCMS. Draft online safety bill. `https://www.gov.uk/government/publications/draft-online-safety-bill`. Accessed on Sep 21, 2021.

[121] D. for DCMS. Draft online safety bill. `https://www.gov.uk/government/publications/draft-online-safety-bill`. Accessed on Sep 21, 2021.

[122] L. Franceschi-Bicchierai. Former nsa chief: I 'would not support' encryption backdoors. `https://www.vice.com/en/article/8qxwda/former-nsa-chief-strongly-disagrees-with-current-nsa-chief-on-encryption`, 2015.

[123] P. Francis, S. P. Eide, and R. Munz. Diffix: High-utility database anonymization. In *Annual Privacy Forum*, pages 141–158. Springer, 2017.

[124] P. Francis, S. Probst-Eide, P. Obrok, C. Berneanu, S. Juric, and R. Munz. Extended diffix. *arXiv preprint arXiv:1806.02075*, 2018.

[125] B. Fraser and J. Wooton. A proposed method for confidentialising tabular output to protect against differencing. *Monographs of Official Statistics: Work Session on Statistical Data Confidentiality*, pages 299–302, 2005.

[126] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.

[127] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 17–32, 2014.

[128] K. Gadiya. FIFA 19 complete player dataset, 2019. `https://www.kaggle.com/datasets/karangadiya/fifa19`.

[129] A. Gadotti, F. Houssiau, L. Rocher, B. Livshits, and Y.-A. de Montjoye. When the signal is in the noise: Exploiting diffix's sticky noise. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1081–1098, 2019.

[130] P. Gainza, F. Sverrisson, F. Monti, E. Rodola, D. Boscaini, M. Bronstein, and B. Correia. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17(2):184–192, 2020.

[131] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 619–633, 2018.

[132] J. Gao, Q. Ping, and J. Wang. Resisting re-identification mining on social graph data. *World Wide Web*, 21:1759–1771, 2018.

[133] J. Gao, Y.-C. Zhang, and T. Zhou. Computational socioeconomics. *Physics Reports*, 817:1–104, 2019.

[134] GIFCT. Gifct transparency report july 2020. `https://gifct.org/wp-content/uploads/2020/10/GIFCT-Transparency-Report-July-2020-Final.pdf`, 2020. Accessed on June 9, 2021.

[135] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[136] V. Gligorijević, P. D. Renfrew, T. Kosciolek, J. K. Leman, D. Berenberg, T. Vatanen, C. Chandler, B. C. Taylor, I. M. Fisk, H. Vlamakis, et al. Structure-based protein function prediction using graph convolutional networks. *Nature communications*, 12(1):3168, 2021.

[137] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[138] S. Goldfeder, H. Kalodner, D. Reisman, and A. Narayanan. When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies. *Proceedings on Privacy Enhancing Technologies*, 4:179–199, 2018.

[139] G. Gonzalez, S. Gong, I. Laponogov, K. Veselkov, and M. Bronstein. Graph attentional autoencoder for anticancer hyperfood prediction. *arXiv preprint arXiv:2001.05724*, 2020.

[140] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[141] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[142] G. Greenwald. NSA collecting phone records of millions of Verizon customers daily. *The Guardian*, 2013.

[143] G. G. Gulyás, B. Simon, and S. Imre. An efficient and robust social network deanonymization attack. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, pages 1–11, 2016.

[144] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger. Simple black-box adversarial attacks. In *International Conference on Machine Learning*, pages 2484–2493. PMLR, 2019.

[145] H. Gupta and H. Taneja. Whatsapp has a fake news problem—that can be fixed without breaking encryption. https://www.cjr.org/tow_center/whatsapp-doesnt-have-to-break-encryption-to-beat-fake-news.php, Aug 2018. Accessed on June 6, 2021.

[146] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[147] C. R. Harris et al. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[148] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava. Anonymizing social networks. *Computer science department faculty publication series*, page 180, 2007.

[149] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[150] X. He, J. Jia, M. Backes, N. Z. Gong, and Y. Zhang. Stealing links from graph neural networks. In *USENIX Security Symposium*, pages 2669–2686, 2021.

[151] R. Hecht-Nielsen. On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers*, pages 129–135. Elsevier, 1990.

[152] A. Herdağdelen, A. Dow, S. Bogdan, M. Payman, and A. Pompe. Protecting privacy in facebook mobility data during the covid-19 response. *Facebook Research*, 2020.

[153] A. Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[154] A. Hern. Cambridge analytica: how did it turn clicks into votes. *The Guardian*, 6, 2018.

[155] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. V. Pearson, D. A. Stephan, S. F. Nelson, and D. W. Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS Genet*, 4(8):e1000167, 2008.

[156] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[157] House of Commons and the House of Lords - Joint Committee on Human Rights. Human rights and the government's response to covid-19: Digital contact tracing. `https://publications.parliament.uk/pa/jt5801/jtselect/jtrights/343/343.pdf`, 2020.

[158] F. Houssiau, J. Jordon, S. N. Cohen, O. Daniel, A. Elliott, J. Geddes, C. Mole, C. Rangel-Smith, and L. Szpruch. Tapas: a toolbox for adversarial privacy auditing of synthetic data. In *NeurIPS 2022 Workshop on Synthetic Data for Empowering ML Research*.

[159] F. Houssiau, L. Rocher, and Y.-A. de Montjoye. On the difficulty of achieving differential privacy in practice: user-level guarantees in aggregate location data. *Nature communications*, 13(1):29, 2022.

[160] M. Humbert, B. Trubert, and K. Huguenin. A survey on interdependent privacy. *ACM Computing Surveys (CSUR)*, 52(6):1–40, 2019.

[161] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Black-box adversarial attacks with limited queries and information. In *International conference on machine learning*, pages 2137–2146. PMLR, 2018.

[162] INRIA. Publication of the ROBERT (ROBust and privacy-presERving proximity Tracing) protocol. `https://www.inria.fr/en/publication-robert-protocol`, 2020.

[163] M. Jagielski, J. Ullman, and A. Oprea. Auditing differentially private machine learning: How private is private sgd? *Advances in Neural Information Processing Systems*, 33:22205–22216, 2020.

[164] M. Jagielski, S. Wu, A. Oprea, J. Ullman, and R. Geambasu. How to combine membership-inference attacks on multiple updated models. *arXiv preprint arXiv:2205.06369*, 2022.

[165] M. Jawurek, M. Johns, and K. Rieck. Smart metering de-pseudonymization. In *Proceedings of the 27th annual computer security applications conference*, pages 227–236, 2011.

[166] B. Jayaraman and D. Evans. Evaluating differentially private machine learning in practice. In *USENIX Security Symposium*, 2019.

[167] B. Jayaraman and D. Evans. Are attribute inference attacks just imputation? In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1569–1582, 2022.

[168] S. Ji, W. Li, N. Z. Gong, P. Mittal, and R. A. Beyah. On your social network de-anonymizablity: Quantification and large scale evaluation with seed knowledge. In *NDSS*. Citeseer, 2015.

[169] S. Ji, W. Li, M. Srivatsa, J. S. He, and R. Beyah. Structure based data de-anonymization of social networks and mobility traces. In *Information Security: 17th International Conference, ISC 2014, Hong Kong, China, October 12-14, 2014. Proceedings 17*, pages 237–254. Springer, 2014.

[170] J. Jin, E. McMurtry, B. I. Rubinstein, and O. Ohrimenko. Are we there yet? timing and floating-point attacks on differential privacy systems. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 473–488. IEEE, 2022.

[171] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *arXiv:1702.08734*, 2017.

[172] N. Johnson, J. P. Near, and D. Song. Towards practical differential privacy for sql queries. *Proceedings of the VLDB Endowment*, 11(5):526–539, 2018.

[173] P. Kairouz, S. Oh, and P. Viswanath. The composition theorem for differential privacy. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1376–1385, Lille, France, 07–09 Jul 2015. PMLR.

[174] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[175] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*. Springer, 2017.

[176] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard. The megaface benchmark: 1 million faces for recognition at scale. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4873–4882, 2016.

[177] K. Kenthapadi and T. T. Tran. Pripearl: A framework for privacy-preserving analytics and reporting at linkedin. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2183–2191, 2018.

[178] J. Kerl. The TMK+PDQF video-hashing algorithim and the PDQ image hashing algorithm. `https://github.com/facebook/ThreatExchange/blob/master/hashing/hashing.pdf`, 2020. Accessed on June 7, 2021.

[179] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR workshop on fine-grained visual categorization (FGVC)*, volume 2. Citeseer, 2011.

[180] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[181] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[182] J. Kleinberg, C. Papadimitriou, and P. Raghavan. Auditing boolean attributes. *Journal of Computer and System Sciences*, 66(1):244–253, 2003.

[183] M. Koppel, J. Schler, and S. Argamon. Authorship attribution in the wild. *Language Resources and Evaluation*, 45:83–94, 2011.

[184] N. Korula and S. Lattanzi. An efficient reconciliation algorithm for social networks. *arXiv preprint arXiv:1307.1690*, 2013.

[185] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[186] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 (canadian institute for advanced research).

[187] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[188] B. Kulynych, M. Yaghini, G. Cherubin, M. Veale, and C. Troncoso. Disparate vulnerability to membership inference attacks. *Proceedings on Privacy Enhancing Technologies*, 1:460–480, 2022.

[189] A. Kurakin, I. Goodfellow, S. Bengio, et al. Adversarial examples in the physical world, 2016.

[190] A. Lavrenovs and K. Podins. Privacy violations in riga open data public transport system. In *2016 IEEE 4th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, pages 1–6. IEEE, 2016.

[191] D. Lazer, A. Pentland, L. Adamic, S. Aral, A.-L. Barabási, D. Brewer, N. Christakis, N. Contractor, J. Fowler, M. Gutmann, et al. Computational social science. *Science*, 323(5915):721–723, 2009.

[192] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[193] K. Leino and M. Fredrikson. Stolen memories: Leveraging model memorization for calibrated {White-Box} membership inference. In *29th USENIX security symposium (USENIX Security 20)*, pages 1605–1622, 2020.

[194] K. Leino, S. Sen, A. Datta, M. Fredrikson, and L. Li. Influence-directed explanations for deep convolutional networks. In *2018 IEEE international test conference (ITC)*, pages 1–8. IEEE, 2018.

[195] I. Levy and C. Robinson. Principles for a more informed exceptional access debate. `https://www.lawfareblog.com/principles-more-informed-exceptional-access-debate`, 2018.

[196] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 123–134, 2010.

[197] C. Li, G. Miklau, M. Hay, A. McGregor, and V. Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB journal*, 24(6):757–781, 2015.

[198] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd international conference on data engineering*, pages 106–115. IEEE, 2006.

[199] Y. Li, J. Yosinski, J. Clune, H. Lipson, and J. Hopcroft. Convergent learning: Do different neural networks learn the same representations? *arXiv preprint arXiv:1511.07543*, 2015.

[200] Z. Li, Y. Liu, X. He, N. Yu, M. Backes, and Y. Zhang. Auditing membership leakages of multi-exit networks. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1917–1931, 2022.

[201] Z. Li and Y. Zhang. Membership leakage in label-only exposures. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 880–895, 2021.

[202] Q. Liu, G. Wang, F. Li, S. Yang, and J. Wu. Preserving privacy with probabilistic indistinguishability in weighted social networks. *IEEE Transactions on Parallel and Distributed Systems*, 28(5):1417–1429, 2016.

[203] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. *arXiv:1611.02770*, 2016.

[204] N. Locascio. Black-box attacks on perceptual image hashes with gans. https://towardsdatascience.com/black-box-attacks-on-perceptual-image-hashes-with-gans-cc1be11f277, 2018. Accessed on June 9, 2021.

[205] Y. Long, L. Wang, D. Bu, V. Bindschaedler, X. Wang, H. Tang, C. A. Gunter, and K. Chen. A pragmatic approach to membership inferences on machine learning models. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 521–534. IEEE, 2020.

[206] F. Lu, J. Munoz, M. Fuchs, T. LeBlond, E. Zaresky-Williams, E. Raff, F. Ferraro, and B. Testa. A general framework for auditing differentially private machine learning. *arXiv preprint arXiv:2210.08643*, 2022.

[207] S. Luo, F. Morone, C. Sarraute, M. Travizano, and H. A. Makse. Inferring personal economic status from social network location. *Nature Communications*, 8(1):15227, 2017.

[208] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.

[209] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[210] T. Man, H. Shen, S. Liu, X. Jin, and X. Cheng. Predict anchor links across social networks via an embedding approach. In *Ijcai*, volume 16, pages 1823–1829, 2016.

[211] J. Mayer. Content moderation for end-to-end encrypted messaging. *Princeton University*, 2019.

[212] F. McSherry. Statistical inference considered harmful. *See https://github.com/frankmcsherry/blog/blob/master/posts/2016-06-14. md*, 2016.

[213] F. D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30, 2009.

[214] M. Meeus, S. Jain, and Y.-A. de Montjoye. Concerns about using a digital mask to safeguard patient privacy. *Nature Medicine*, 29(7):1658–1659, 2023.

[215] S. Mehnaz, S. V. Dibbo, R. De Viti, E. Kabir, B. B. Brandenburg, S. Mangard, N. Li, E. Bertino, M. Backes, E. De Cristofaro, et al. Are your sensitive attributes private? novel model inversion attribute inference attacks on classification models. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4579–4596, 2022.

[216] P. Melo, J. Messias, G. Resende, K. Garimella, J. Almeida, and F. Benevenuto. Whatsapp monitor: A fact-checking system for whatsapp. *Proceedings of the International AAAI Conference on Web and Social Media*, 13:676–677, Jul 2019.

[217] I. Memon, Q. Ali, A. Zubedi, and F. A. Mangi. Dpmm: dynamic pseudonym-based multiple mix-zones generation for mobile traveler. *Multimedia Tools and Applications*, 76:24359–24388, 2017.

[218] L. Meunier, J. Atif, and O. Teytaud. Yet another but more efficient black-box adversarial attack: tiling and evolution strategies. *arXiv:1910.02244*, 2019.

[219] M. K. Mıhçak and R. Venkatesan. New iterative geometric methods for robust perceptual image hashing. In *ACM Workshop on Digital Rights Management*, pages 13–21. Springer, 2001.

[220] I. Mironov. On significance of the least significant bits for differential privacy. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 650–661, 2012.

[221] M. Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[222] V. Mnih, N. Heess, A. Graves, et al. Recurrent models of visual attention. *Advances in neural information processing systems*, 27, 2014.

[223] F. Monti, F. Frasca, D. Eynard, D. Mannion, and M. M. Bronstein. Fake news detection on social media using geometric deep learning. *arXiv preprint arXiv:1902.06673*, 2019.

[224] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *International workshop on artificial intelligence and statistics*, pages 246–252. PMLR, 2005.

[225] S. U. Nabar, K. Kenthapadi, N. Mishra, and R. Motwani. A survey of query auditing techniques for data privacy. In *Privacy-Preserving Data Mining*, pages 415–431. Springer, 2008.

[226] A. Narayanan, H. Paskov, N. Z. Gong, J. Bethencourt, E. Stefanov, E. C. R. Shin, and D. Song. On the feasibility of internet-scale author identification. In *2012 IEEE Symposium on Security and Privacy*, pages 300–314. IEEE, 2012.

[227] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125. IEEE, 2008.

[228] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *2009 30th IEEE symposium on security and privacy*, pages 173–187. IEEE, 2009.

[229] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[230] M. Nasr, S. Songi, A. Thakurta, N. Papernot, and N. Carlin. Adversary instantiation: Lower bounds for differentially private machine learning. In *2021 IEEE Symposium on security and privacy (SP)*, pages 866–882. IEEE, 2021.

[231] NCMEC. Ncmec's statement regarding end-to-end encryption. `https://www.missingkids.org/blog/2019/post-update/end-to-end-encryption`, Mar 2019. Accessed on June 6, 2021.

[232] J. Near. Differential privacy at scale: Uber and berkeley collaboration. In *Enigma 2018 (Enigma 2018)*, 2018.

[233] Y. Nesterov and V. Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

[234] B. News. Cambridge Analytica parent firm SCL Elections fined over data refusal. *BBC News*, 2019.

[235] J. Neyman and E. S. Pearson. Ix. on the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933.

[236] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[237] NHSX. Documentation relating to the Beta of the NHS COVID-19 app. `https://github.com/nhsx/COVID-19-app-Documentation-BETA/`, 2020.

[238] S. Nilizadeh, A. Kapadia, and Y.-Y. Ahn. Community-enhanced de-anonymization of online social networks. In *Proceedings of the 2014 acm sigsac conference on computer and communications security*, pages 537–548, 2014.

[239] Norwegian Institute of Public Health. NIPH stops collection of personal data in Smittestopp. `https://www.fhi.no/en/news/2020/niph-stops-collection-of-personal-data-in-smittestopp/`, 2020.

[240] E. Notification. Cryptography specification. v1. 2 april 2020. apple & google.

[241] K. Numpacharoen and A. Atsawarungruangkit. Generating correlation matrices based on the boundaries of their coefficients. *PloS one*, 7(11):e48902, 2012.

[242] O2. Our privacy policy. `https://www.o2.co.uk/termsandconditions/privacy-policy`, 2020. accessed on 16/12/2020.

[243] A. Oehmichen, S. Jain, A. Gadotti, and Y.-A. de Montjoye. Opal: High performance platform for large-scale privacy-preserving location data analytics. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1332–1342. IEEE, 2019.

[244] H. Office. Interim code of practice on online child sexual exploitation and abuse (accessible version). https://www.gov.uk/government/publications/online-harms-interim-codes-of-practice/interim-code-of-practice-on-online-child-sexual-exploitation-and-abuse-accessible- Dec 2020. Accessed on June 6, 2021.

[245] P. Ohm. Broken promises of privacy: Responding to the surprising failure of anonymization. *UCLA l. Rev.*, 57:1701, 2009.

[246] C. M. O'Keefe, S. Haslett, D. Steel, and R. Chambers. Table builder problem-confidentiality for linked tables. 2008.

[247] OpenAI. Introducing chatgpt.

[248] OpenSafely. Secure analytics platform for nhs electronic health records, 2021. Accessed on Nov 8, 2021.

[249] R. Overdorf and R. Greenstadt. Blogs, twitter feeds, and reddit comments: Cross-domain authorship attribution. *Proc. Priv. Enhancing Technol.*, 2016(3):155–171, 2016.

[250] PACT. Pact: Private automated contact tracing. https://pact.mit.edu/, 2020.

[251] M. Pandey, M. Fernandez, F. Gentile, O. Isayev, A. Tropsha, A. C. Stern, and A. Cherkasov. The transformational role of gpu computing and deep learning in drug discovery. *Nature Machine Intelligence*, 4(3):211–221, 2022.

[252] N. Papernot et al. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[253] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman. Sok: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 399–414. IEEE, 2018.

[254] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908*, 2018.

[255] N. Papernot and T. Steinke. Hyperparameter tuning with renyi differential privacy. *arXiv preprint arXiv:2110.03620*, 2021.

[256] N. Papernot, A. Thakurta, S. Song, S. Chien, and Ú. Erlingsson. Tempered sigmoid activations for deep learning with differential privacy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9312–9321, 2021.

[257] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[258] P. Patel, W. Barr, P. Dutton, A. Little, B. Blair, India, and Japan. Internation statement: End-to-end encryption and public safety. `https://www.gov.uk/government/publications/international-statement-end-to-end-encryption-and-public-safety`, Oct 2020. Accessed on June 6, 2021.

[259] P. Pedarsani, D. R. Figueiredo, and M. Grossglauser. A bayesian method for matching two similar graphs without seeds. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1598–1607. IEEE, 2013.

[260] C. Peersman, J. T. Llanos, C. May-Chahal, R. McConville, P. D. Chowdhury, and E. De Cristofaro. Towards a framework for evaluating csam prevention and detection tools in the context of end-to-end-encryption environments: a case study. 2023.

[261] J. C. Pinheiro and D. M. Bates. Unconstrained parametrizations for variance-covariance matrices. *Statistics and computing*, 6:289–296, 1996.

[262] J. Powar and A. R. Beresford. Sok: Managing risks of linkage attacks on data privacy. *Proceedings on Privacy Enhancing Technologies*, 2:97–116, 2023.

[263] President's Council of Advisors on Science and Technology. Big data and privacy: A technological perspective. `https://bigdatawg.nist.gov/pdf/pcast_big_data_and_privacy_-_may_2014.pdf`, 2014.

[264] A. Pyrgelis, C. Troncoso, and E. De Cristofaro. Knock knock, who's there? membership inference on aggregate location data. *NDSS*, 2018.

[265] Pyrgelis, Apostolos. On location, time, and membership: Studying how aggregate location data can harm users' privacy. https://www.benthamsgaze.org/2018/10/02/on-location-time-and-membership-studying-how-aggregate-location-data-can-harm-user 2018.

[266] Python Core Team. Python: A dynamic, open source programming language. https://www.python.org/, 2021.

[267] S. Qummar, F. G. Khan, S. Shah, A. Khan, S. Shamshirband, Z. U. Rehman, I. A. Khan, and W. Jadoon. A deep learning ensemble approach for diabetic retinopathy detection. *Ieee Access*, 7:150530–150539, 2019.

[268] M. Redmond. UCI Repository of machine learning databases: Communities and Crime Unnormalized Data Set, 2011. https://archive.ics.uci.edu/ml/datasets/communities+and+crime+unnormalized.

[269] J. C. S. Reis, P. Melo, K. Garimella, and F. Benevenuto. Can whatsapp benefit from debunked fact-checked stories to reduce misinformation? *Harvard Kennedy School Misinformation Review*, 2020.

[270] A. Rekognition. Moderating content. https://docs.aws.amazon.com/rekognition/latest/dg/moderation.html.

[271] K. Ren, T. Zheng, Z. Qin, and X. Liu. Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346–360, 2020.

[272] U. M. L. Repository. Adult data set, 1996.

[273] U. M. L. Repository. Census-income (kdd) data set, 2000.

[274] U. M. L. Repository. Insurance company benchmark (coil 2000) data set, 2000.

[275] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[276] C. Riederer, Y. Kim, A. Chaintreau, N. Korula, and S. Lattanzi. Linking users across domains with location data: Theory and validation. In *Proceedings of the 25th international conference on world wide web*, pages 707–719, 2016.

[277] Y. Rinott, C. M. O'Keefe, N. Shlomo, and C. Skinner. Confidentiality and Differential Privacy in the Dissemination of Frequency Tables. *Statistical Science*, 33(3):358 – 385, 2018.

[278] L. Rocher, J. M. Hendrickx, and Y.-A. de Montjoye. Estimating the success of re-identifications in incomplete datasets using generative models. *Nature communications*, 10(1):1–9, 2019.

[279] R. Rogers, A. R. Cardoso, K. Mancuhan, A. Kaura, N. Gahlawat, N. Jain, P. Ko, and P. Ahammad. A members first approach to enabling linkedin's labor market insights at scale. *arXiv preprint arXiv:2010.13981*, 2020.

[280] O. Russakovsky et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[281] A. Sablayrolles, M. Douze, C. Schmid, Y. Ollivier, and H. Jégou. White-box vs black-box: Bayes optimal strategies for membership inference. In *International Conference on Machine Learning*, pages 5558–5567. PMLR, 2019.

[282] A. Salem, A. Bhattacharya, M. Backes, M. Fritz, and Y. Zhang. {Updates-Leak}: Data set inference and reconstruction attacks in online learning. In *29th USENIX security symposium (USENIX Security 20)*, pages 1291–1308, 2020.

[283] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246*, 2018.

[284] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[285] P. Sapiezynski, A. Stopczynski, D. D. Lassen, and S. Lehmann. Interaction data from the copenhagen networks study. *Scientific Data*, 6(1):315, 2019.

[286] J. Saramäki, E. A. Leicht, E. López, S. G. Roberts, F. Reed-Tsochas, and R. I. Dunbar. Persistence of social signatures in human communication. *Proceedings of the National Academy of Sciences*, 111(3):942–947, 2014.

[287] Y. Sari, A. Vlachos, and M. Stevenson. Continuous n-gram representations for authorship attribution. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 267–273, 2017.

[288] K. Sastry, D. Goldberg, and G. Kendall. Genetic algorithms. In *Search methodologies*, pages 97–125. Springer, 2005.

[289] S. Schoen. What location tracking looks like. *Electronic Frontier Foundation*, Mar 2011.

[290] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[291] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.

[292] R. Schwartz, O. Tsur, A. Rappoport, and M. Koppel. Authorship attribution of micro-messages. In *Proceedings of the 2013 Conference on empirical methods in natural language processing*, pages 1880–1891, 2013.

[293] M. Seshadri, S. Machiraju, A. Sridharan, J. Bolot, C. Faloutsos, and J. Leskove. Mobile call graphs: beyond power-law and lognormal distributions. In *Proceedings of the 14th*

ACM SIGKDD international conference on Knowledge discovery and data mining, pages 596–604, 2008.

[294] C. Shao, G. L. Ciampaglia, O. Varol, K.-C. Yang, A. Flammini, and F. Menczer. The spread of low-credibility content by social bots. *Nature communications*, 9(1):1–9, 2018.

[295] Y. Shao, J. Liu, S. Shi, Y. Zhang, and B. Cui. Fast de-anonymization of social networks with structural information. *Data Science and Engineering*, 4:76–92, 2019.

[296] K. Sharad. Change of guard: The next generation of social graph de-anonymization attacks. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, pages 105–116, 2016.

[297] K. Sharad and G. Danezis. An automated social graph de-anonymization technique. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 47–58, 2014.

[298] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 acm sigsac conference on computer and communications security*, pages 1528–1540, 2016.

[299] T. Sharma and M. Bashir. Use of apps in the covid-19 response and the loss of privacy protection. *Nature Medicine*, 26(8):1165–1167, 2020.

[300] X. Shen, S. Diamond, Y. Gu, and S. Boyd. Disciplined convex-concave programming. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 1009–1014. IEEE, 2016.

[301] Y. Shi, S. Wang, and Y. Han. Curls & whey: Boosting black-box adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6519–6527, 2019.

[302] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[303] M. Singh. Whatsapp is now delivering roughly 100 billion messages a day. https://social.techcrunch.com/2020/10/29/whatsapp-is-now-delivering-roughly-100-billion-messages-a-day/, Oct 2020. Accessed on June 6, 2021.

[304] M. Singh. Signal's brian acton talks about exploding growth, monetization and whatsapp data-sharing outrage. https://social.techcrunch.com/2021/01/12/signal-brian-acton-talks-about-exploding-growth-monetization-and-whatsapp-data-sha Jan 2021. Accessed on June 6, 2021.

[305] S. P. Singh and M. Jaggi. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33:22045–22055, 2020.

[306] V. K. Singh, L. Freeman, B. Lepri, and A. S. Pentland. Predicting spending behavior using socio-mobile features. In *2013 International Conference on Social Computing*, pages 174–179. IEEE, 2013.

[307] M. Sklar. Fonctions de répartition à n dimensions et leurs marges. In *Annales de l'ISUP*, volume 8, pages 229–231, 1959.

[308] S. Song, K. Chaudhuri, and A. D. Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE global conference on signal and information processing*, pages 245–248. IEEE, 2013.

[309] N. Spier, S. Nekolla, C. Rupprecht, M. Mustafa, N. Navab, and M. Baust. Classification of polar maps from cardiac perfusion imaging with graph-convolutional neural networks. *Scientific reports*, 9(1):7569, 2019.

[310] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[311] M. Srivatsa and M. Hicks. Deanonymizing mobility traces: Using social network as a side-channel. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 628–637, 2012.

[312] T. Stadler, B. Oprisanu, and C. Troncoso. Synthetic data – anonymisation groundhog day. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1451–1468, Boston, MA, Aug. 2022. USENIX Association.

[313] E. Stamatatos. Authorship verification: a review of recent advances. *Research in Computing Science*, 123:9–25, 2016.

[314] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life*, 15(2):185–212, 2009.

[315] J. Stempel. NSA's phone spying program ruled illegal by appeals court. *Reuters*, 2015.

[316] J. Su, D. V. Vargas, and K. Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[317] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[318] A. Suri and D. Evans. Formalizing and estimating distribution inference risks. *Proceedings on Privacy Enhancing Technologies*, 4:528–551, 2022.

[319] J. Svoboda, J. Masci, and M. M. Bronstein. Palmprint recognition via discriminative index learning. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 4232–4237. IEEE, 2016.

[320] L. Sweeney. Weaving technology and policy together to maintain confidentiality. *The Journal of Law, Medicine & Ethics*, 25(2-3):98–110, 1997.

[321] L. Sweeney. Simple demographics often identify people uniquely. *Health (San Francisco)*, 671(2000):1–34, 2000.

[322] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv:1312.6199*, 2013.

[323] N. Tatro, P.-Y. Chen, P. Das, I. Melnyk, P. Sattigeri, and R. Lai. Optimizing mode connectivity via neuron alignment. *Advances in Neural Information Processing Systems*, 33:15300–15311, 2020.

[324] The White House, Office of the Press Secretary. Statement by the president. `https://obamawhitehouse.archives.gov/the-press-office/2013/06/07/statement-president`, 2013.

[325] S. Thielman. Surveillance reform explainer: can the FBI still listen to my phone calls? *The Guardian*, 2015.

[326] Thorn. Testing different image hash functions. `https://content-blockchain.org/research/testing-different-image-hash-functions/`. Accessed on June 8, 2021.

[327] A. Tockar. Riding with the stars: Passenger privacy in the nyc taxicab dataset. *Neustar Research, September*, 15:6, 2014.

[328] F. Tramer and D. Boneh. Differentially private learning needs better features (or much more data). *arXiv preprint arXiv:2011.11660*, 2020.

[329] D.-A. Trégouët, P. Ducimetiere, V. Bocquet, S. Visvikis, F. Soubrier, and L. Tiret. A parametric copula model for analysis of familial binary data. *The American Journal of Human Genetics*, 64(3):886–893, 1999.

[330] TriNetX. Explore. discover. connect. `https://trinetx.com/`, 2023.

[331] C. Troncoso et al. Decentralized privacy-preserving proximity tracing. `https://github.com/DP-3T/documents/blob/master/DP3T%20White%20Paper.pdf`, 2020.

[332] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei. Demystifying membership inference attacks in machine learning as a service. *IEEE Transactions on Services Computing*, 14(6):2073–2089, 2019.

[333] V. Tudor, M. Almgren, and M. Papatriantafilou. A study on data de-pseudonymization in the smart grid. In *Proceedings of the Eighth European Workshop on System Security*, pages 1–6, 2015.

[334] V. Tudor, M. Almgren, and M. Papatriantafilou. The influence of dataset characteristics on privacy preserving methods in the advanced metering infrastructure. *Computers & Security*, 76:178–196, 2018.

[335] United States Census Bureau. Census bureau sets key parameters to protect privacy in 2020 census results. `https://www.census.gov/newsroom/press-releases/2021/2020-census-key-parameters.html`, 2021.

[336] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[337] M. Veale, R. Binns, and L. Edwards. Algorithms that remember: model inversion attacks and data protection law. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2133):20180083, 2018.

[338] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[339] K. Verhelst, K. Aouasti, A. M. Freilich, J. Soors, G. V. Burre, N. Gilson, K. Verduykt, S. Mahdi, and C. Fonck. Proposition de loi relative à l'utilisation d'applications numériques de traçage de contacts par mesure de prévention contre la propagation du coronavirus covid-19 parmi la population. `https://www.lachambre.be/FLWB/PDF/55/1251/55K1251001.pdf`, 2020.

[340] K. Veselkov, G. Gonzalez, S. Aljifri, D. Galea, R. Mirnezami, J. Youssef, M. Bronstein, and I. Laponogov. Hyperfoods: Machine intelligent mapping of cancer-beating molecules in foods. *Scientific reports*, 9(1):9237, 2019.

[341] P. Virtanen et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[342] A. Voyez, T. Allard, G. Avoine, P. Cauchois, E. Fromont, and M. Simonin. Unique in the smart grid-the privacy cost of fine-grained electrical consumption data. *arXiv preprint arXiv:2211.07205*, 2022.

[343] P. Waldmann. On the use of the pearson correlation coefficient for model evaluation in genome-wide prediction. *Frontiers in genetics*, 10:899, 2019.

[344] G. K. Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[345] G. Wang, Q. Liu, F. Li, S. Yang, and J. Wu. Outsourcing privacy-preserving social networks to a cloud. In *2013 Proceedings IEEE INFOCOM*, pages 2886–2894. IEEE, 2013.

[346] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[347] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009.

[348] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

[349] Q. Xiao. Calculating correlation coefficient for gaussian copula. *arXiv preprint arXiv:1608.00738*, 2016.

[350] Q. Xiao, Y. Chen, C. Shen, Y. Chen, and K. Li. Seeing is not believing: Camouflage attacks on image scaling algorithms. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 443–460, 2019.

[351] K. Yang, J. Yau, L. Fei-Fei, J. Deng, and O. Russakovsky. A study of face obfuscation in imagenet. *arXiv:2103.06191*, 2021.

[352] L. Yartseva and M. Grossglauser. On the performance of percolation graph matching. In *Proceedings of the first ACM conference on Online social networks*, pages 119–130, 2013.

[353] J. Ye, A. Maddi, S. K. Murakonda, V. Bindschaedler, and R. Shokri. Enhanced membership inference attacks against machine learning models. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 3093–3106, 2022.

[354] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, pages 268–282. IEEE, 2018.

[355] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. pages 268–282, 2018.

[356] J. S. Yoo, A. Thaler, L. Sweeney, and J. Zang. Risks to patient privacy: a re-identification of patients in maine and vermont statewide hospital data. *J Technol Sci*, 2018:2018100901, 2018.

[357] A. Yousefpour, I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao, et al. Opacus: User-friendly differential privacy library in pytorch. *arXiv preprint arXiv:2109.12298*, 2021.

[358] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International conference on machine learning*, pages 7252–7261. PMLR, 2019.

[359] S. Zanella-Béguelin, L. Wutschitz, S. Tople, A. Salem, V. Rühle, A. Paverd, M. Naseri, and B. Köpf. Bayesian estimation of differential privacy. *arXiv preprint arXiv:2206.05199*, 2022.

[360] H. Zang and J. Bolot. Anonymization of location data does not work: A large-scale measurement study. In *Proceedings of the 17th annual international conference on Mobile computing and networking*, pages 145–156, 2011.

[361] C. Zauner. Implementation and benchmarking of perceptual image hash functions. 2010.

[362] C. Zauner, M. Steinebach, and E. Hermann. Rihamark: perceptual image hash benchmarking. In *Media watermarking, security, and forensics III*, volume 7880. International Society for Optics and Photonics, 2011.

[363] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[364] W. Zhang, K. Shu, H. Liu, and Y. Wang. Graph neural networks for user identity linkage. *arXiv preprint arXiv:1903.02174*, 2019.

[365] W. Zhang, S. Tople, and O. Ohrimenko. Leakage of dataset properties in multi-party machine learning. In *USENIX Security Symposium*, pages 2687–2704, 2021.

[366] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *2008 IEEE 24th International Conference on Data Engineering*, pages 506–515. IEEE, 2008.

[367] H. Zhou, Z. Deng, Y. Xia, and M. Fu. A new sampling method in particle filter based on pearson correlation coefficient. *Neurocomputing*, 216:208–215, 2016.

[368] J. Zhou, Y. Chen, C. Shen, and Y. Zhang. Property inference attacks against gans. *arXiv preprint arXiv:2111.07608*, 2021.

[369] M. Zitnik, M. Agrawal, and J. Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.

[370] L. Zou, L. Chen, and M. T. Özsu. K-automorphism: A general framework for privacy preserving network publication. *Proceedings of the VLDB Endowment*, 2(1):946–957, 2009.

[371] S. Zuboff. *The age of surveillance capitalism: The fight for a human future at the new frontier of power: Barack Obama's books of 2019*. Profile books, 2019.