Leakage Abuse Attacks on Encrypted Columns using Lp-optimization

Chelsea Ling Xinyi[1], Ananya Kharbanda[2], Ruth Ng Ii-Yung[3]

[1] Raffles Girls' School, 2 Braddell Rise, Singapore 318871
[2] Raffles Institution, 1 Raffles Institution Ln, Singapore 575954
[3] DSO National Laboratories, 20 Science Park Dr, Singapore 118230

chelsea.ling07@rafflesgirlssch.edu.sg
24YANAN481Z@student.ri.edu.sg
niiyung@dso.org.sg

Abstract. Our work studies a class of leakage abuse attacks to address the Single-Column Full-Information Leakage Abuse Attack (SF-LAA) problem called the Lp-optimization LAA. Each such attack is defined for some p>1. In this work, we implement three such attacks (for p=1,2,3) using Python, by reducing the problem first to Lp-norm minimization, then reducing that to the Linear Sum Assignment Problem (LSAP). Alongside the three Lp-optimization attacks, we implement Frequency Analysis as an LAA and compare the effectiveness of the four LAAs by calculating two different scores.

Keywords: leakage abuse attacks; structured encryption; assignment problems.

1 Introduction

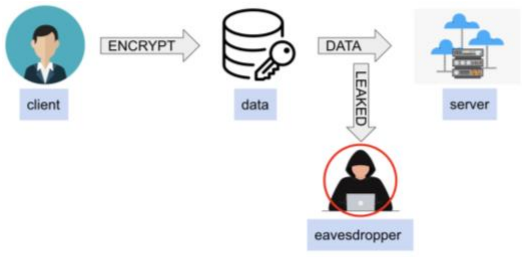


Fig. 1. Leakage being leaked to eavesdropper while client queries their encrypted data

In the 21st century, most of our confidential information such as healthcare records, credit card numbers etc. is stored on the cloud. While some may think that deterministically encrypting data and storing it on the cloud is completely secure, the encrypted tables may be accessed by eavesdroppers when the client performs `SELECT` queries.

The eavesdropper can then derive the frequencies of each ciphertext, called the leakage. Attackers can perform leakage abuse attacks such as Lp-optimization and Frequency Analysis on this leakage, and obtain confidential information. A proposed solution to address this weakness is "CryptDB"[1] which uses a Deterministic Encryption (DET) system, to support SQL `SELECT` queries on relational data. However, DET has its own weakness, hence the aim of our project is to demonstrate the weakness of DET via cryptanalysis, and prove experimentally that Lp-optimization is only optimal when p>1.

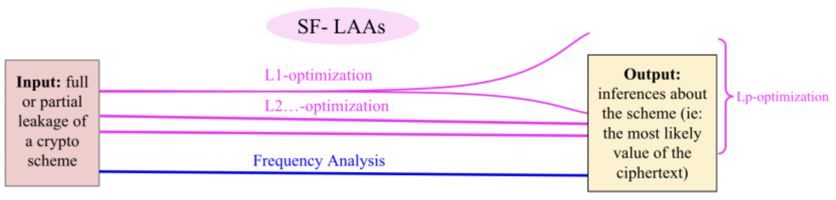
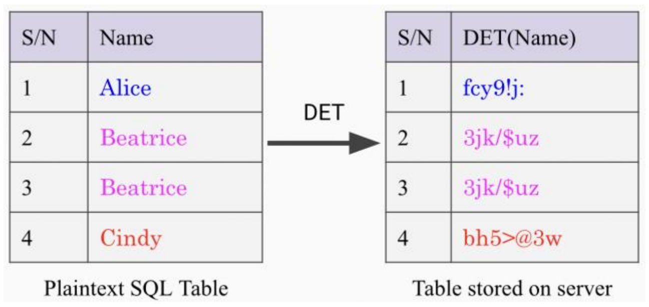

SF- LAAs
Input: full or partial leakage of a crypto scheme
L1-optimization
L2…-optimization
Frequency Analysis
Output: inferences about the scheme (ie: the most likely value of the ciphertext)
Lp-optimization

Fig. 2. A visual representation of the LAAs in this report. A line connecting the red "input" box to the yellow "output" box represents a LAA.

In Fig. 2, the Leakage Abuse Attacks (LAAs) in the diagram are directly connected to the "input" box and "output" box which means that for those types of attacks, all inputs would return the most likely said output. However, the route in the diagram with the fork shows how some inputs to L1-optimization return an incorrect output. However, L2 and L3-optimization do not face this problem and are always equivalent to Frequency Analysis. In this report, we refer to four LAAs—L1-optimization, L2-optimization, L3-optimization and Frequency Analysis.

2 Preliminaries

2.1 Deterministic encryption (DET)



S/N	Name
1	Alice
2	Beatrice
3	Beatrice
4	Cindy

Plaintext SQL Table

DET

S/N	DET(Name)
1	fcy9!j:
2	3jk/$uz
3	3jk/$uz
4	bh5>@3w

Table stored on server

Fig. 3: Example of a table being encrypted with DET.

DET encrypts each value in a column so that one can efficiently perform `SELECT` queries by replacing each plaintext value with DET(Value) while keeping the structure of the table the same. This way, the client can search for a value simply by querying DET(Value). For example, in Fig. 3, each value in the "Name" column of the plaintext SQL Table is replaced with the DET(Value), but the structure of the table stored on the server and the S/N column remains the same. If the client wants to select all "Beatrice" rows, the server can be queried as DET(Beatrice) = "3jk/$uz". As seen in Fig. 3, the 3 different types of names are encrypted in different ways. "Beatrice", which occurs twice, is encrypted the same way both times. This shows that despite the encryption, the same value will be encrypted the same way throughout the column, allowing an attacker to obtain the frequency of the occurrence of each value, also known as ciphertext frequencies, as a leakage.
For n different values, the vector **c** would be a string of length n, where each number corresponds to the frequency of the occurrence of each value in the column. The

ciphertext frequencies are then represented in a vector form. For example, the ciphertext frequencies in vector form for Fig. 3. are **c** = (1,2,1).

2.2 Reduction

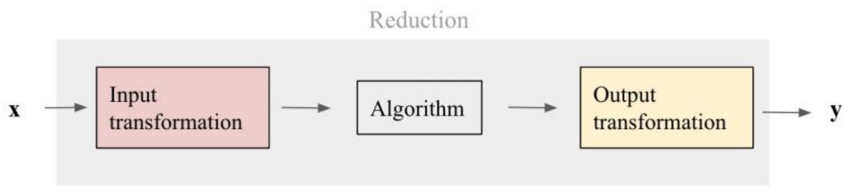


Fig. 4. In a reduction algorithm, an input and an output transformation must be defined, both of which are algorithms.

As seen in Fig. 4, the input transformation converts the input to a format that the algorithm can take, and the output transformation converts the output back to the desired format. For example, **x** may have been converted to another data type before being converted back to a vector **y**. Reductions are used to convert variables into a simpler form that we can use to solve.

2.3 Auxiliary information

Auxiliary information is information related to, but not the same as the ciphertext data, which is available to the attacker and may also be public information. For example, the ciphertext data could be a name list of girls in a school, while the auxiliary data might be the most popular baby girl names in their birth year. This information can be public or leaked from a database (i.e. from a previous attack).

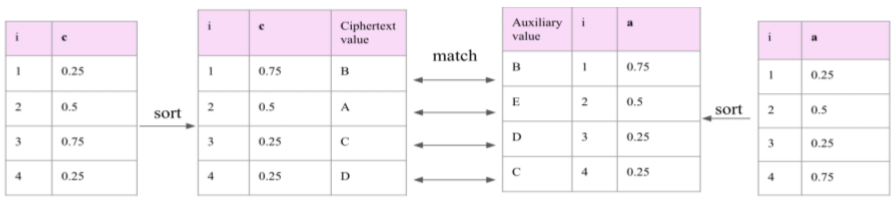
2.4 Leakage Abuse Attacks (LAAs)

The goal of a Leakage Abuse Attack is to use the leakage explicitly allowed by a cryptographic scheme in order to reveal/obtain information about input data. In this report, the datasets used only consist of one column each, and the frequencies of every value are known to the eavesdropper. The goal of an LAA is to output a high-probability mapping. The attack is *optimal* if it returns the maximal-probability mapping. The input of an LAA in this report should be in vector form and the output should be a function. Formally,

$$\vec{a},\vec{c} \Longrightarrow LAA \Longrightarrow f^* = \begin{matrix} f:\{1,\dots,n\} \\ \to \{1,\dots,n\} \end{matrix} \ \Pr[f] = f\prod_{i=1}^{n} a_i^{c^{f(i)}}$$

We use this equation to summarize the properties of an LAA. The inputs of an LAA are **a** and **c**, which are the auxiliary and ciphertext vectors. There are n values of **a** and **c**. The output of an LAA is f^*, which is, ideally, a high probability of f.

3 Frequency Analysis



i	c
1	0.25
2	0.5
3	0.75
4	0.25

i	c	Ciphertext value
1	0.75	B
2	0.5	A
3	0.25	C
4	0.25	D

Auxiliary value	i	a
B	1	0.75
E	2	0.5
D	3	0.25
C	4	0.25

i	a
1	0.25
2	0.5
3	0.25
4	0.75

Fig. 5: A Frequency Analysis attack on a dataset in a Searchable Encryption application.

Frequency Analysis is an LAA which we used to approach the problem. We input the auxiliary probability distribution **a** = {a_1,...,a_n} and ciphertext frequencies **c** = {c_1,...,c_n}. We assume that **a** is a good estimate of the actual distribution of ciphertexts, since the auxiliary and ciphertext datasets are related. First, **a** and **c** are sorted in descending order. Frequency Analysis then matches the first value of **c** to the first value of **a**, and so on.
Formally, for i = 1,…,n, we match the ith largest ciphertext frequency with the ih largest auxiliary percentage. f^* would be the mapping of the value of the ith largest ciphertext frequency and the value of ih largest auxiliary percentage. When there are two equal values, we order them randomly.
However, 2 things may cause Frequency Analysis to perform undesirably. First, when the auxiliary information does not perfectly reflect the actual ciphertext distribution, some matchings will naturally be wrong. Second, when there are many equal values, the algorithm is not able to do better than randomly ordering and matching the values, since the ciphertext frequencies are identical. This may cause some matchings to be incorrect.

4 Lp-optimization

This section will describe 3 LAAs. This is called the Lp family, where p ∈ Z, p > 0, and each value of p defines a different LAA.

$$\begin{bmatrix} |\boldsymbol{a_1} - \boldsymbol{c_1}|^{\boldsymbol{p}} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & |\boldsymbol{a_i} - \boldsymbol{c_j}|^{\boldsymbol{p}} \end{bmatrix}$$

Fig. 6: Cost matrix of values $|a_1$-$c_1|^p$ to $|a_i$-$c_j|^p$ 4.1 Overview of Lp-optimization

Lp-optimization is designed with the assumption that the closer the auxiliary and ciphertext frequencies are to each other, the higher the probability of getting the matching correct. We observe that a high-probability mapping would assign high c_j to high a_i and vice-versa, so that (a_1, … a_n) and ($c_{f(1)}$, … $c_{f(n)}$) are highly correlated. Hence, we match the scale of a, c then, for each frequency f, we consider how "close" a_i and c_j are from each other. Our LAA returns the f corresponding to the "closest" pair of a_i and c_j. his "closeness" is measured using the Lp-norm:

$$\sum_{i=1}^{n} |\boldsymbol{a_i} - \boldsymbol{c_j}|^{\boldsymbol{p}} {}^{\frac{1}{p}}$$

Hence, the lower the Lp-norm, the higher the probability of getting the matching correct. In Fig. 7, the inputs are the ciphertext frequencies **c**, and the auxiliary probability distribution **a**. The 1st input transformation converts the inputs into the correct "formats" that can be accepted by the main algorithm. The 2nd input

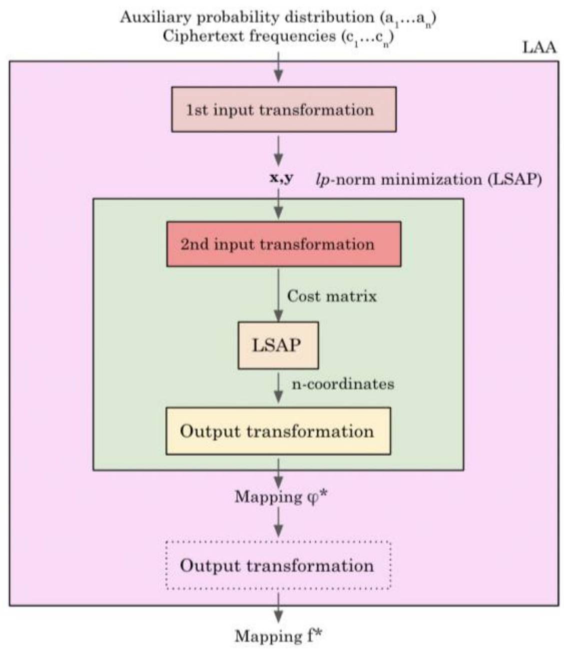


Fig. 7: Flowchart showing the Lp-optimization algorithm, an example of a reduction.

transformation takes $|a_i-c_j|^p$ for $i = 1,\ldots,n$, $j = 1,\ldots,n$ and reshapes it into a cost matrix (Fig. 7). The output of the 2nd input transformation, the cost matrix, then goes through the Linear Sum Assignment Problem algorithm (LSAP). The output from LSAP would be the mappings of the corresponding values inside the cost matrix, ϕ*. Since ϕ* is the function that returns the minimal Lp-norm, it is essentially equal to f^*, so we do not need to transform it again. In the Lp-norm equation, each value of p represents a different LAA attack variant, hence we run Lp-optimization three times, with each variant of p=1,2 and 3 respectively.

4.2 Linear Sum Assignment Problem (LSAP)

The LSAP algorithm is used to solve the argmin of a $n \times n$ cost matrix, $C = (c_{ij})$. The goal is to match each row to a different column such that the sum of the corresponding values is minimized. We want to select n elements of C so that there is exactly one element in each row and one in each column and the sum of the corresponding costs is a minimum. First, the values are plotted in a cost matrix. The minimum value of each row is then subtracted from each value in said row and the process is repeated for the columns. The optimal mapping is found by crossing out all the zeros using as few lines as possible. If this condition is not met, subtract all the remaining values by the smallest value (among all the remaining values). Once this condition is met, assign each row to a box in the corresponding column with value 0. The optimal assignment would be the sum of the original value of the selected boxes, which would be the combination with the minimum sum.

4.3 Why L1-optimization can be inaccurate

Contrary to L2 and L3-optimization LAAs, L1-optimization LAA does not always return an optimal result. This is because if there are a few results that are equally good, it will pick one at random and we have no way of confirming if it picked correctly. Hence, only L2 and L3-optimization (and Frequency Analysis) return the argmax of *f,* which can be shown by our LAA equation

$$\vec{a},\vec{c} \Longrightarrow L2/L3/FA \Longrightarrow f^* = \underset{\substack{f:\{1,\dots,n\}\\ \to \{1,\dots,n\}}}{argmax} \Pr[f] = \underset{f}{argmax} \prod_{i=1}^{n} a_i^{c\,f(i)}$$

In order to visualise a scenario where L1-optimization is not always optimal, we illustrate the counterexample shown below.

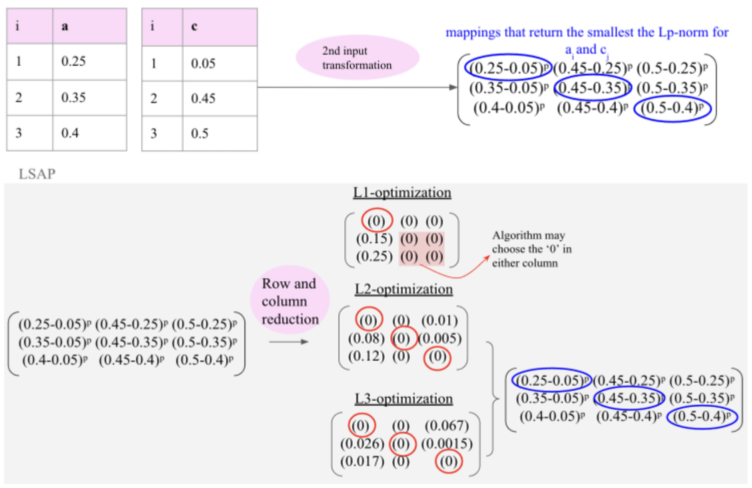


Fig. 8: Sample values of **a** and **c** being transformed into a cost matrix and undergoing a row and column reduction in LSAP.

Fig. 8 shows how there exists a combination of '0' positions such that there is only one '0' per row and column selected in L2-optimization and L3-optimization, but not L1-optimization.

5 Results

To compare the differences between L1-optimization, L2-optimization, L3-optimization and Frequency Analysis, we run 3 attack setups on 15 columns across 2 real-world datasets. The datasets in our experiment include the National Inpatient Sample (NIS) 2018 and 2019 [2] from the Healthcare Cost and Utilization Project (HCUP), and voter data from Florida and Ohio [3]. We use 1 million rows of each dataset.
We measure the success of an LAA by calculating its R-Score and V-Score. The R-Score is the number of ciphertext rows that was guessed correctly divided by the total number of ciphertext rows. The V-Score is equal to the number of ciphertext values mapped correctly divided by the total number of ciphertext values. To ensure that n is constant, we take the first 2000 ciphertext and auxiliary values with the highest frequency. If the length of **c** > length of **a**, we append "0" values to **a**. If length of **a** > **c**, we equate the length of **a** to **c** by truncating **a**.

Table 1. Attack setups and their corresponding columns

Attack setup	**HCUP 2018-HCUP 2019**	**Florida-Ohio voter data**	**Florida voter data-HCUP 2019**
Columns	Age, Neonatal Age Indicator, Admission Month, Admission day is a weekend, Died during hospitalization, DRG in effect on discharge date, MDC in effect on discharge date, IDC-10-CM Diagnosis 1, Number of days from admission to I10_PR1, Indicator of Sex, Race	First names, Last names	Indicator of sex, Race

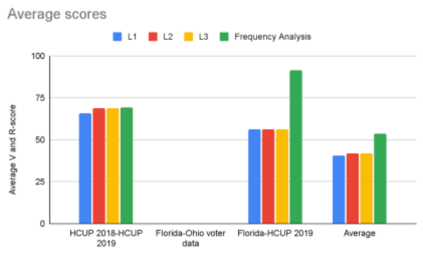
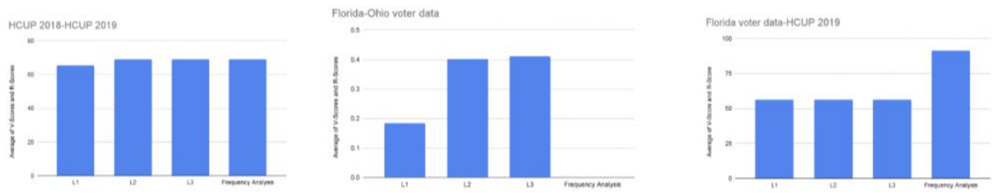


Fig. 9a. Overview of the average V and R-Scores of all 4 LAAs



(from left to right)
Fig. 9b: HCUP 2018 - HCUP 2019 **Fig. 9c:** Florida voter data - Ohio voter data **Fig. 9d:** Florida voter data - HCUP 2019

Fig. 9: Averages of the V-Score and R-Score of each LAA for the respective attack setups. The final score shown in all the graphs in Fig. 9 is calculated by taking the average of the R-Score and V-Score of every column for each attack setup.

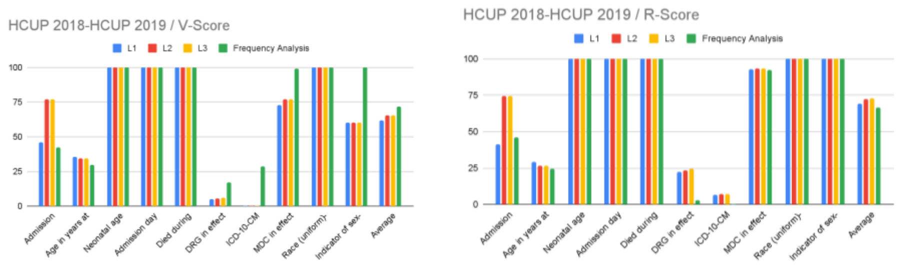


Fig. 10: V-Scores and R-Scores of each column in the HCUP 2018-HCUP 2019 attack setup.

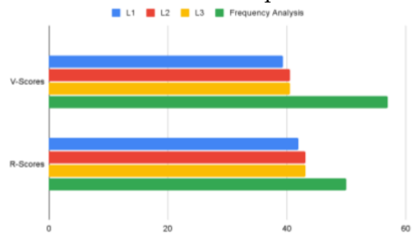


Fig. 11: Comparison between the average V-Score and R-Score of all 4 LAAs.

5.1 General observations

From the above graphs, we can see that our attack works on the attack setups we previously defined. On average, we are able to retrieve around half of the ciphertext values. Since some values contribute to a large portion of the total rows, we managed to retrieve close to 75% of the total rows. This means that our attacks were fairly successful, which may have negative real-world impacts should the attacks be carried out by real adversaries. We will further break down our results in the following sections. From Fig. 11, for the attacks on the HCUP 2018-HCUP 2019, we were able to obtain full information about the "Neonatal Age", "Admission Day" and "Race" through Frequency Analysis.

5.2 Comparison between LAAs

We compare the averages of the V-Scores and R-Scores for all columns of each LAA. Overall, the scores of L2-optimization and L3-optimization are highly correlated, and are on average 42% higher than that of L1-optimization. This agrees with our prior theoretical knowledge, proving that p>1 minimizes the Lp-norm, thus Lp-optimization where p>1 is able to get more rows correct, increasing its score.

Nonetheless, in Fig. 9d, L1-optimization, L2-optimization and L3-optimization score the same for the Florida voter data-HCUP 2019 attack setup. This also proves our theory that some inputs to L1-optimization may return the correct outputs, while some may not. The case in Fig. 9d could be one that happened to return the correct outputs for all values.

There seems to be an anomaly in Fig. 9c, where Frequency Analysis scored 0. This is because Frequency Analysis, in theory, performs better if the auxiliary frequencies and ciphertext probability distribution are closely correlated and there are few values with the same frequency. The first and last names of people in Florida and Ohio may be drastically different, and may have many instances of the same number of people in that dataset having a certain name. Thus, this could be why Frequency Analysis was not able to get any of the rows correct in the Florida-Ohio voter data setup.

We observe that Frequency Analysis generally scores higher than all 3 variants of Lp-optimization. This is especially obvious for the non-numerical columns of the V-Score in Fig. 9. We hypothesize that this could be due to its implementation in the code, which artificially inflated the values. We further hypothesize that this does not occur in Lp-optimization as the LSAP library does not use sorting unlike

Frequency Analysis. When we use the `sorted()` function [4] in a Python dictionary, when the frequencies of 2 values are the same, it will proceed to sort the values alphabetically. This means that values with equal frequencies in both the auxiliary and ciphertext datasets will be ordered in the same way, hence there will be more correct rows. For example, if "Bob" = 0.1 and "Alice" = 0.1 in both **a** and **c**, "Alice" will always be sorted above "Bob" in both **a** and **c**. However, in a real-world scenario where the data is encrypted, even if the data is sorted, it only has a 50/50 chance of matching the rows correctly as the attacker would not know the plaintext of the ciphertext data.
The trends in Fig. 11 correspond to our hypothesis, as a lower V-Score indicates that the LAA only got values with higher frequencies correct, as seen in the Lp-optimization scores. For Frequency Analysis, its higher V-Score with an R-Score similar to the Lp-optimization LAAs indicates that it got most of the values with higher frequencies and some of the values with lower frequencies correct. Since the values with equal frequencies usually only have 1 row (i.e. the "rarer" values), this means that Frequency Analysis might indeed have gotten many of the values with lower frequencies correct due to sorting.

5.3 Comparison between attack setups

In Fig. 9a, the HCUP attack setup does the best, while the voter data setup does the worst. This is probably because most of the columns we ran in the HCUP datasets had fewer values and larger distribution of frequencies. For example, the column "Admission day is a weekend" only had values 0, 1, -9, with most of the rows being 1. Whereas for the voter datasets, the "first name" and "last name" columns that we ran had a lot of different values and the distribution of frequencies was very narrow. In other words, there were many unique names that only had one person in the entire dataset. This could have caused all the attacks to perform more badly for the voter data.

5.4 Comparison between V-Score and R-Score

In Fig. 11, we discover that the R-Score for all 4 LAAs is 37-47% higher than its V-Score. This could be because there are more correct rows than values, especially in case of more "popular" values. For example, the name (value) Michael has 30000 rows, while the name "Robin" has 2 rows. If the LAA gets "Michael" right and "Robin" wrong, the V-Score will be ½ = 0.5, while the R-Score will be 30000/30002 = 0.99.

6 Related Work

[5] was the first to provide experimental results from Frequency Analysis and Lp-optimization, in response to high-profile data breaches. It did not include optimality proofs. [6] solves the problem using bipartite matching. This solution is proven to be optimal and equivalent to the effectiveness of Frequency Analysis. [7] then provides the mathematical proof of Frequency Analysis's optimality in solving SF-LAA.

7 Conclusion

In this paper, we present how deterministic encryption may be an avenue for eavesdroppers to retrieve confidential data. We present four attacks which were given in prior work, two of which have been shown to be optimal. We then additionally show that the fourth strategy is optimal under certain conditions.
First, we introduced LAAs such as Lp-optimization and Frequency Analysis, their inputs and outputs, and proceeded to measure their success in the form of R-Scores and V-Scores. We briefly explained why using variants of Lp-optimization, where p>1, would minimize the Lp-norm and thus have a higher probability of returning accurate mappings.
In all, this paper documents various attack scenarios that are extremely plausible in the real world, given that some important data may already have been encrypted with DET, which could pose a security risk. It also allows us to experimentally prove that Lp-optimization returns more successful results when p>1, and that Frequency Analysis works the best among all leakage abuse attacks. From our results, we conclude that deterministic encryption is a weak type of encryption and not a good solution, as an adversary can extract useful information and exploit it for their use. For example, in a real-world scenario, this information can be used by insurance companies who can charge higher premiums for certain races. Another example of exploiting these results would be invading people's personal privacy and leaking controversial healthcare information about certain important public figures. Future work may extend the results in the paper when comparing Lp-optimization and Frequency Analysis to other LAAs.

8 Acknowledgements

We would like to thank our mentors, Dr Ruth Ng and Mr Ng Wei Cheng, and our RGS teacher-mentor, Ms Carmen Hoo, for providing us with valuable guidance.

9 References

[1] Popa, et al. *CryptDB: Protecting Confidentiality with Encrypted Query Processing*, SOSP 2011, 1 Oct. 2011, https://dspace.mit.edu/handle/1721.1/74107.
[2] Agency for Healthcare Research and Quality. *NIS Archive. Healthcare Cost and Utilization Project (HCUP). https://hcup-us.ahrq.gov/db/nation/nis/nisarchive.jsp*
[3] *Voter Demographic Data*, www.leonvotes.gov/Records-Data-Maps/Voter-Demographic-Data
[4] Dalke, Hettinger. *Sorting HOW TO*, Python Software Foundation, docs.python.org/3/howto/sorting.html
[5] Naveed, et al. *Inference Attacks on Property-Preserving Encrypted Databases*, ACM-CCS '15, 12 Oct 2015, https://www.microsoft.com/en-us/research/wp-content/uploads/2017/02/edb.pdf
[6] Bindschaedler, et al. *The Tao of Inference in Privacy-Protected Databases*, VLDB Endowment, 1 Jul. 2018, https://eprint.iacr.org/2017/1078.pdf
[7] Lacharité, Paterson. *A note on the optimality of frequency analysis vs. -optimization*, 30 Nov. 2015, https://eprint.iacr.org/2015/1158.pdf