Week 3 Inference Attacks on Encrypted Databases

Learning outcomes

In this week's tutorial, we will review the property-preserving encryption (PPE), its advantages and disadvantages. We will conduct case studies with real-world public datasets and show how to implement inference attacks against PPE in practice. We hope you can fully understand how PPE is used in encrypted database systems, and the assumptions and the methodology of the inference attacks against PPE.

Task 1: Review Questions

Q1. Please explain how PPE is applied for building encrypted databases? What are the advantages of PPE

Q2. What information leakage can a database server obtain if deterministic encryption and order-preserving encryption are applied?

Task 2: Inference Attacks on PPE

1. Frequency analysis on deterministic encryption:

In this task, we will first apply a symmetric deterministic encryption (DTE) on records of a relational database column, and then launch a frequency analysis to break the confidentiality of that deterministically-encrypted column.

Step 1: The dataset used in this task, **transaction_ds.csv**, contains the synthesised transactions of 100 accounts by ANZ [1]. The attached **task1_cipher_gen.py** script encrypts every data column in the dataset by using an AES-CBC cipher.

1. *Please run the script and describe/explain your observation after obtaining the cipher output in **det_enc.ds.csv***

Step 2: In this step, we launch the frequency analysis attack to uncover the ciphertext versions of accounts in **det_enc.ds.csv.** In detail, the attack requires two inputs. Those are, an auxiliary data for which the attacker can always consult public information sources via public statics or prior version of data breaches, and the encrypted dataset. For simplicity, we consider the auxiliary data perfectly matches the raw version, i.e., **transaction_ds.csv.** For the second input, i.e, **det_enc.ds.csv**, the attacker can directly obtain since the encrypted data is maintained by untrusted cloud providers or a database server, which can be accessed by the attacker.

1. *Please run* ***task1_frequency_analysis.py*** *script* to *investigate how the attack works.*

2. *Please explain why a ciphertext can match to multiple raw accounts in an output* ***map.txt****.*

   Solution: In short, the attack deterministically matches each ciphertext to possibly matching raw data on the same frequency ranking. In practice, the attack's success also depends on how well-correlated the auxiliary data is with the plaintext column.

2. Frequency analysis on order-preserving encryption:

In this task, we demonstrate the sorting attack to a dense OPE-encrypted column. We note that the column is dense ($\delta = 1$) if it contains at least a fraction $\delta = 1$ of its plaintext space.

Step 1: The dataset used in this task, **sales_ds.csv**, contains the historical sales of a supermarket company transactions which has recorded in 3 different branches for 3 months data [2]. of 100 accounts by ANZ [1]. The attached **cipher_gen.py** script encrypts every data column in the dataset by using OPE for a given cipher's output range.

1. *Please run the script and describe/explain your observation after obtaining the cipher output in* ***ope_enc.ds.csv***

Step 2: In this step, we launch the sorting attack to infer the mapping between the cipher and the possible plaintext. We note that the attack does not require the auxiliary information of the frequency occurrence of the data. Instead, it simply performs the mapping for which the cipher text and the plaintext are on the same unique ranking.

1. *Please run* ***sorting_attack.py*** *script* to *investigate how the attack works.*

3. L_P-Optimisation attacks on PPE

In this section, we introduce a basic attack of the l_p- optimisation attacks against DTE-encrypted columns. The fundamental idea of the attack is to identify a mapping that minimises the l_p-distance between the histograms of the ciphertext dataset **c** and plaintext dataset **z** [3]. We assume that those datasets belong to the ciphertext space C_n and plaintext space M_n of the DTE scheme, and $|C_n| = |M_n|$. Formally, the attack is defined as:

l_p-Optimisation(c,z):

```
    1: compute $\psi \leftarrow Hist(c)$
    2: compute $\pi \leftarrow Hist(z)$
```

3: output arg $arg\ min_{X \in P_n} \leftarrow || \psi - X \cdot \pi ||_p$

where P_n is the set of $n \times n$ permutation matrices, for each one is denoted as X.

For $p = 1$ (i.e., l_1-optimisation attack), the step 3 is a linear sum assignment problem (LSAP) on a 1-dimensional datasets **c** and **z** (see the **account** column in **transaction_ds.csv** and **det_enc_ds.csv).** The LSAP can be formulated as:

minimise $\sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij}$

subject to $\sum_{i=1}^{n} X_{ij} = 1,\ 1 \le j \le |C_n|$ //sum of all weights on a row =1

$\sum_{j=1}^{n} X_{ij} = 1,\ 1 \le i\ \le\ |C_n|$ //sum of all weights on a column =1

$X_{ij} \in \{0, 1\}, \quad 1 \le i, j\ \le\ |C_n|$

With $p = 1$, the cost matrix $C = C_{ij}$ presents the absolute difference in frequency, where $C_{ij} = |\pi_i - \psi_j|$

1. *Please run the script **l_1_analysis.py** to obtain the cipher output in **map.txt***
2. *What is the difference between the frequency analysis attack in* Section II.1 *and* l_1 *-optimisation attack?*

Home work:

Q1: Think about how the l_p- optimisation attack work with $p = 2$.

Q2: Think about how to mitigate inference attacks.

Reference:

[1] ANZ synthesis dataset,
https://www.kaggle.com/bhatvikas/anz-synthesised-transaction-dataset

[2] Supermarket transaction dataset,
https://www.kaggle.com/aungpyaeap/supermarket-sales?select=supermarket_sales+-+Sheet1.csv

[3] Muhammad Naveed, Seny Kamara, and Charles V. Wright. 2015. Inference Attacks on Property-Preserving Encrypted Databases. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15). Association for Computing Machinery, New York, NY, USA, 644–655. DOI:https://doi.org/10.1145/2810103.2813651