FIT 5124 Advance Topics in Security (S1 2021)

Assignment 1: Total marks 100

Due on 3 April 11:59:59 am (Firm!)

1 Overview

The objective of this assignment is to assess the learning outcomes in cryptographic techniques and protocols for searching over encrypted data. Specifically, the tasks will evaluate your knowledge of security analysis of searchable symmetric encryption (SSE), count attacks against SSE, countermeasures for SSE schemes, and inference attacks against property-preserving encryption.

2 Submission Policy

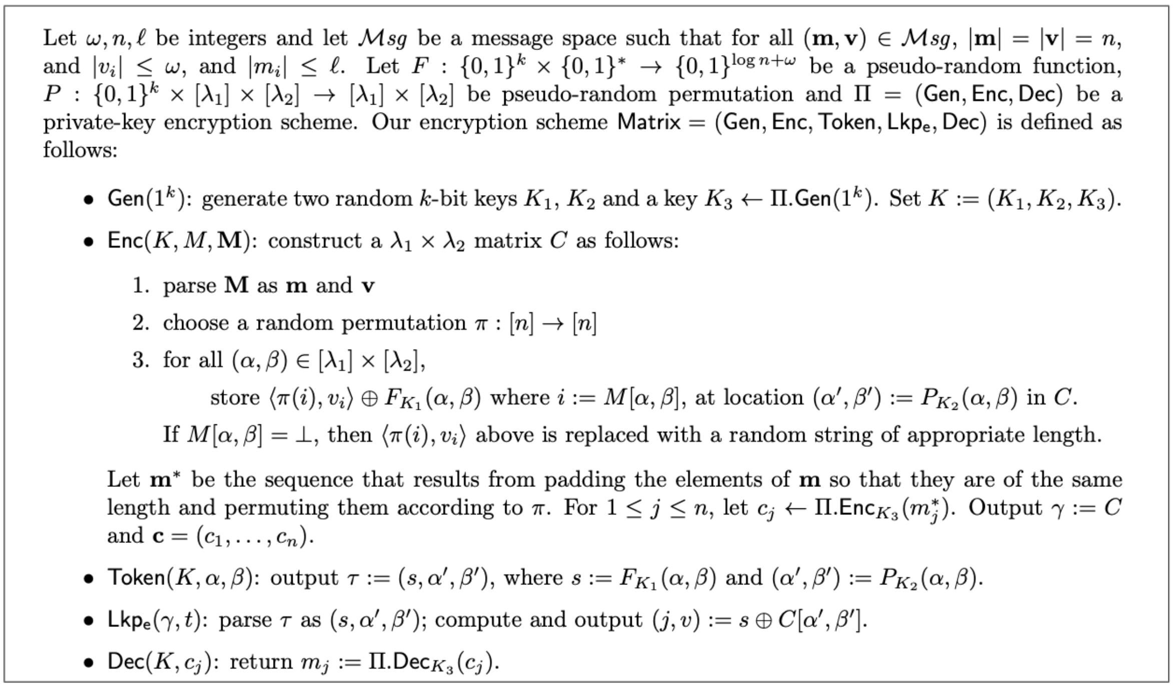
You need to submit a report (one single PDF file) to describe what you have done and what you have observed with screen shots whenever necessary; you also need to provide explanation or codes to the observations that are related to the tasks. In your report, you are expected to answer all the questions listed in this manual. Typeset your report into .pdf format (make sure it can be opened with Adobe Reader) and name it as the format:
[Your Name]-[Student ID]-FIT5124-Assignment1,
HarryPotter-12345678-FIT5124-Assignment1.pdf.

All source code if required should be embedded in your report. Please upload the PDF file to Moodle. Note that the assignment is due on **3 April, Saturday, 11:59:59 am (No extension)**.

Late submission penalty: 10-point deduction per day. If you require a special consideration, the application should be submitted at least three days in advance via Monash Connect (https://www.monash.edu/connect). Zero tolerance on plagiarism: If you are found cheating, penalties will be applied, i.e., a zero grade for the unit. The demonstration video is also used to detect/avoid plagiarism. University policies can be found at https://www.monash.edu/students/academic/policies/academic-integrity

3 Security Analysis of Searchable Symmetric Encryption (30 Marks)

Perform the formal security analysis for the scheme in the paper of "Structured Encryption and Controlled Disclosure" https://eprint.iacr.org/2011/010.pdf (see the screenshot below). More detailed construction and notations can be found in the paper.



Let ω, n, ℓ be integers and let $\mathcal{M}sg$ be a message space such that for all $(\mathbf{m}, \mathbf{v}) \in \mathcal{M}sg$, $|\mathbf{m}| = |\mathbf{v}| = n$, and $|v_i| \leq \omega$, and $|m_i| \leq \ell$. Let $F : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^{\log n + \omega}$ be a pseudo-random function, $P : \{0,1\}^k \times [\lambda_1] \times [\lambda_2] \rightarrow [\lambda_1] \times [\lambda_2]$ be pseudo-random permutation and $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a private-key encryption scheme. Our encryption scheme $\mathsf{Matrix} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Lkp_e}, \mathsf{Dec})$ is defined as follows:

- $\mathsf{Gen}(1^k)$: generate two random k-bit keys K_1, K_2 and a key $K_3 \leftarrow \Pi.\mathsf{Gen}(1^k)$. Set $K := (K_1, K_2, K_3)$.
- $\mathsf{Enc}(K, M, \mathbf{M})$: construct a $\lambda_1 \times \lambda_2$ matrix C as follows:
  1. parse $\mathbf{M}$ as $\mathbf{m}$ and $\mathbf{v}$
  2. choose a random permutation $\pi : [n] \rightarrow [n]$
  3. for all $(\alpha, \beta) \in [\lambda_1] \times [\lambda_2]$,
     store $\langle \pi(i), v_i \rangle \oplus F_{K_1}(\alpha, \beta)$ where $i := M[\alpha, \beta]$, at location $(\alpha', \beta') := P_{K_2}(\alpha, \beta)$ in C.
     If $M[\alpha, \beta] = \perp$, then $\langle \pi(i), v_i \rangle$ above is replaced with a random string of appropriate length.

  Let $\mathbf{m}^*$ be the sequence that results from padding the elements of $\mathbf{m}$ so that they are of the same length and permuting them according to π. For $1 \leq j \leq n$, let $c_j \leftarrow \Pi.\mathsf{Enc}_{K_3}(m_j^*)$. Output $\gamma := C$ and $\mathbf{c} = (c_1, \ldots, c_n)$.
- $\mathsf{Token}(K, \alpha, \beta)$: output $\tau := (s, \alpha', \beta')$, where $s := F_{K_1}(\alpha, \beta)$ and $(\alpha', \beta') := P_{K_2}(\alpha, \beta)$.
- $\mathsf{Lkp_e}(\gamma, t)$: parse τ as (s, α', β'); compute and output $(j, v) := s \oplus C[\alpha', \beta']$.
- $\mathsf{Dec}(K, c_j)$: return $m_j := \Pi.\mathsf{Dec}_{K_3}(c_j)$.

Figure 1: An associative structured encryption scheme for matrices.

3.1 Security Definition (10 Marks)

Please give the simulation-based security definition for the above scheme regarding **non-adaptive** security.

3.2 Security Analysis (20 Marks)

Please perform the formal security analysis based on the security definition and the construction. Hint: leakage function definition (5 marks), steps on how to simulate the security game (10 marks), explanation on how the simulated view is indistinguishable from the real view (5 marks).

4 Count Attacks against Searchable Symmetric Encryption (40 marks)

4.1 Case study I: given a document set {d1, d2, d3, d4, d5, d6}, and d1 = {w1, w2, w3, w4, w6}, d2 = {w1, w2, w3, w6}, d3 = {w1, w2, w4, w6}, d4 = {w1, w4, w6}, d5 = {w4, w5, w6}, d6 = {w4, w6}, and assume that the adversary knows the above information about the document set. Then the adversary starts to launch count attacks for the document set encrypted by an SSE scheme. He observed the following query response from the server.

q	**eids**					
q1	e1	e2	e3	e5	e6	
q2	e3	e4				
q3	e1	e2	e3	e4		
q4	e2	e3	e4			
q5	e1	e2	e3	e4	e5	e6
q6	e6					

Note that e1,..., e6 are the permuted document IDs and the adversary does not know the mapping between them and the original ones.

Q: Please explain how to recover the underlying keyword of each query step by step, and write down the mapping between queries and keywords (10 marks).

4.2 Case study II: given a document set {d1, d2, d3, d4, d5, d6}, and d1 = {w1, w2, w3, w4, w6}, d2 = {w2, w3, w6}, d3 = {w1, w3, w4, w6}, d4 = {w1, w4, w6}, d5 = {w4, w5, w6}, d6 = {w5}, and assume that the adversary knows the above information about the document set. Then the adversary starts to launch count attacks for the document set encrypted by an SSE scheme. He observed the following query response from the server.

q	**eids**					
q1	e2	e3	e4	e5	e6	
q2	e2	e3	e5	e6		
q3	e1	e3				

q4	e4	e5				
q5	e2	e5	e6			
q6	e4	e5	e6			

Note that e1,..., e6 are the permuted document IDs and the adversary does not know the mapping between them and the original ones.

Q: Please explain how to recover the underlying keyword of each query step by step, and write down the mapping between queries and keywords. Can the count attack recover all queries? If not, please explain the reason. (15 marks)

4.3 Case study III: given a document set {d1, d2, d3, d4, d5, d6}, and d1 = {w1, w2, w3, w6}, d2 = {w2, w3, w6}, d3 = {w1, w2, w3, w4, w6}, d4 = {w1, w3, w4, w6}, d5 = {w5, w6}, d6 = {w5, w6}, and assume that the adversary knows the above information about the document set. Then the adversary starts to launch count attacks for the document set encrypted by an SSE scheme. He observed the following query response from the server.

q	**eids**					
q1	e1	e5	e6			
q2	e1	e2	e6			
q3	e3	e4				
q4	e1	e2	e3	e4	e5	e6
q5	e2	e5	e6			
q6	e1	e2	e5	e6		

FIT 5124 Advance Topics in Security (S1 2021)

Q: Please explain how to recover the underlying keyword of each query step by step, and write down the mapping between queries and keywords. Can the count attack recover all queries? If not, please explain the reason. (15 marks)

5 Padding Countermeasures in Searchable Symmetric Encryption (20 Marks)

The primary countermeasure for the above count attack is padding solution. In this task, you are given the attached source code and data:

- *Inverted_index_5000*: This binary file contains the inverted index of the top 5000 most frequent keywords in the real Enron email dataset (https://www.cs.cmu.edu/~enron/).
- *frequency_5000.csv*: This .csv file details the (keyword, frequency) in the above index. This .csv file is already sorted in the descending order.
- *app.py* and *utilities.py*. The current *app.py* script reads the above inverted index into a Python dictionary data structure, namely `inverted_dict`. Then, using `inverted_dict[keyword]` will give you the list of document identifiers mapping to that `keyword`. You can review these two scripts for more details.

The first padding approach (Approach #1) is to add more padding (w,id') pairs for each keyword w such that all the keywords have the same number of real and padding pairs, where id' is a padding document identifier. In this way, all keywords will have the result lengths the same as the result length of the most frequent keyword when the untrusted server executes the search operation in SSE scheme.

The second padding solution (Approach #2) is to add more padding (w,id') pairs to w such that the query's result length of w is a multiplication of an integer. For example, the current frequency of w is 6, then it needs 4 padding (w,id') pairs if the multiplication of 5 is used.

The above two approaches can be found in [4].

The third padding solution (Approach #3) is to only pad keywords that have the similar frequencies into the same result length. As a result, this approach will minimise the padding overhead. However, this approach first requires defining the clusters of keywords in advance. Then, we can identify the most frequent keyword in each cluster and pad the remaining keywords in the same cluster to that length. More details about this approach can be found in [5].

Q1: Please develop the padding solution in Approach #1 and identify the padding overhead. Note that the padding overhead is the ratio between the total number of real and padding pairs in the inverted index over the real number of pairs. Add the screenshot of your python code and the result into your report **(5 marks)**

Q2: Please develop the padding solution in Approach #2 and identify the padding overhead if the multiplication of 100 is used. Add the screenshot of your python code and the result into your report**(5 marks)**

Q3: Please develop the padding solution in Approach #3 and identify the padding overhead. Note that you are given the cluster configuration in the *app.py*. Add the screenshot of your python code and the result into your report **(5 marks)**

For example,

```
cluster_points_256 =
[392,648,904,1160,1416,1672,1928,2184,2440,2696,2952,3208,3464,3720
,3976,4232,4488,4744]
```

denotes that the 1st keyword to the 392nd keyword (inclusive) in *frequency_5000.csv* will be in the first cluster, and the 4745th keyword to the 5000th keyword (inclusive) will be in the last cluster. There are 19 clusters in this setting. With this configuration, there are at least 256 keywords in each cluster (i.e., cluster size).

Another configuration is that,

```
cluster_points_512 = [904,1416,1928,2440,2952,3464,3976,4488]
```

will have at least 512 keywords in the same cluster.

Q4: Compare the padding overhead between the above Approach #1, Approach #2, and Approach #3 with cluster sizes 256 and 512. Add the screenshot of your python code and the result into your report **(5 marks)**

6 Inference Attacks against Order-preserving Encryption (10 Marks)

As introduced in Week 3, sorting attacks is applicable to uncover dense OPE-encrypted column ($\delta = 1$). In that setting, a ciphertext contains at least a fraction of $\delta = 1$ plaintext(s) to perform a direct mapping on the same sorting order. However, the attack becomes less applicable for low-density OPE-encrypted columns if the fraction is less than 1. Intuitively, the plaintext space (i.e., auxiliary dataset) known by the attacker who is launching the inference attack might be smaller than the underlying plaintext space of the ciphertext space.

To address the limitation of the sorting attacks when dealing with low-density OPE-encrypted columns , Naveed et. al. [1] introduced cumulative attack. In short, the attack combines the ordering with the frequencies to increase the ability to match a plaintext to a ciphertext. For example, if a ciphertext matches 90% of its ciphertext in the encrypted column, then the mapping should match that ciphertext to a plaintext that also matches 90% of its plaintext. To do this, the attacker uses empirical cumulative distribution function [2] (ECDF, or simply called CDF) to evaluate the probability of a real-valued ciphertext. Formally, the attack is defined as:

Cumulative-Atk(c, z): Given an OPE-encrypted column *c* over the ciphertext space **C** and the auxiliary plaintext dataset *z* over the plaintext space **M**, the adversary computes:

1. compute $\psi \leftarrow Hist(c)$ and $\varphi \leftarrow CDF(c)$
2. compute $\pi \leftarrow Hist(z)$ and $\mu \leftarrow CDF(z)$
3. output $arg\ min_{X \in P} \sum_{i=1}^{|M|} (|\psi_i - X_i \cdot \pi| + |\varphi_i - X_i \cdot \mu|)$

where $\psi$and π are the histograms of *c* and *z,* respectively; $\varphi$and μ are the CDFs of *c* and *z,* respectively; P is the set of $|C| \times |C|$ permutation matrices.

The step 3 can be formulated as a linear sum assignment problem (LSAP) as below:

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

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

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

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

where the cost matrix Cgives the cost for mapping a plaintext $m_j$to a cipher text $c_i$as the sum of the mismatch in the frequencies plus the mismatch in cumulative frequencies: $C_{ij} = |\psi_i - \pi_j|^2 + |\varphi_i - \mu_j|^2$

In the attached material, you are given an OPE-encrypted column in ***ope_enc_covid_age.csv***, and the auxiliary plaintext dataset in ***auxiliary.csv***. The dataset is extracted from the Mexican government's COVID19 dataset [3]. The ***cumulative_attack.py*** contains the instructions how to perform Steps 1-3.

FIT 5124 Advance Topics in Security (S1 2021)

Q1. Complete the Step 1 by calculating the histograms of the cipher *c* and the plaintext *z*. Please add your script screenshot to your report **(2.5 marks)**

Q2. Complete the Step 2 by calculating the empirical CDFs of *c* and *z*. Please add your script screenshot to your report. **(2.5 marks)**

Q3. In Step 3, compute the cost matrix C and apply the LSAP resolver to identify the optimal weight matrix X. Please add your script screenshot to your report **(2.5 marks)**

Q4. Print the inference mapping between the cipher and the plaintext, and copy it into your assignment report.**(2.5 marks)**

Reference:

[1] 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

[2] CDF and Empirical CDF, https://en.wikipedia.org/wiki/Cumulative_distribution_function#Empirical_distribution_function

[3] COVID-19 patient pre-condition dataset, https://www.kaggle.com/tanmoyx/covid19-patient-precondition-dataset

[4] Cash, D., Grubbs, P., Perry, J., & Ristenpart, T. (2015, October). Leakage-abuse attacks against searchable encryption. In Proceedings of the 22nd ACM SIGSAC conference on computer and communications security (pp. 668-679).

[5] Bost, Raphael, and Pierre-Alain Fouque. "Thwarting Leakage Abuse Attacks against Searchable Encryption-A Formal Approach and Applications to Database Padding." IACR Cryptol. ePrint Arch. 2017 (2017): 1060.