Secure Cloud Computing (202000026)
Q1 2024/2025
Issue Date: 23 September, 2024 **Due Date: 08 October 2024, 23:59 CEST**

Assignment 2: Encrypted Databases and Searchable Symmetric Encryption

Introduction

In this assignment, you will study searchable symmetric encryption (SSE) used to protect outsourced files and encrypted databases. There is one theoretical question on SSE and one practical assignment on encrypted databases.

Submission Guidelines

This assignment will be done in pairs (groups of size 2). You have to hand in a PDF document with your answers and your source code (plain text) on Canvas. Please state your group number and your own name and UT-student number as well as the name and UT-student number of your partner. Do not submit your solution with multiple accounts but use the group submission on Canvas.

Important Do not submit PDFs of hand-written documents (such as scans or photos) – *we will not take handwritten submissions into account.* Instead write your documents using LaTeX, MS-Word or similar, and then save/export as PDF.

1 Proving Information Leakage of an SSE Scheme *(10 Points)*

A searchable symmetric encryption (SSE) scheme for static databases as introduced in the lecture consists of the following algorithms:

$k \leftarrow \mathsf{Gen}(1^\lambda)$: takes the unary encoding of the security parameter λ as input and outputs a secret key k.

$\mathcal{I} \leftarrow \mathsf{BuildIndex}(k, \mathcal{D})$: takes the secret key k and a document collection $\mathcal{D}$ as input and outputs an encrypted search index $\mathcal{I}$.

$\tau_q \leftarrow \mathsf{SearchToken}(k, q)$: takes the secret k and a keyword q as input and outputs a search token τ_q.

$D[q] \leftarrow \mathsf{Search}(\mathcal{I}, \tau_q)$: takes the encrypted index $\mathcal{I}$ and a search token τ_q as input and outputs the corresponding document identifiers matching the search query $\mathcal{D}[q]$.

A common approach for quantifying and proving information leakage for SSE schemes is based on a simulation-based security game and has been introduced by Curtmola et al. [1] as semantic security. Assume you implement a simple searchable encryption scheme based on a deterministic function similar to the approach proposed for CryptDB. For this

assignment assume that keywords can have an arbitrary (but bounded) length and the deterministic mapping is modeled as Pseudorandom Function (PRF) with fixed output length of n bits. Given a PRF $F : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^n$ the construction is summarized as follows:

$k \leftarrow \mathsf{Gen}(1^\lambda)$: takes a unary encoding of the security parameter λ as input and outputs a secret key k for the PRF F.

$\mathcal{I} \leftarrow \mathsf{BuildIndex}(k, \mathcal{D})$: takes the secret key k and a document collection $\mathcal{D}$ of size d as input. For each document $D_i \in \mathcal{D}$ consisting of keywords $\{p_1, \ldots, p_{n_i}\}$ it computes the encrypted index entry C_i for document D_i as follows:

- Encrypt each keyword p_j using the PRF F and secret key k and output $\gamma_j = F(k, p_j)$.
- Permute all encrypted keywords $\{\gamma_1, \ldots, \gamma_{n_i}\}$ in C_i randomly (you can ignore this permutation in your security considerations).

Finally, the encrypted search index consisting of all encrypted index entries $\mathcal{I} = \{C_1, C_2, \ldots, C_d\}$ is returned.

$\tau_q \leftarrow \mathsf{SearchToken}(k, q)$: takes the secret key k and a keyword q as input and outputs the search token $\tau_q = F(k, q)$.

$D[q] \leftarrow \mathsf{Search}(\mathcal{I}, \tau_q)$: takes the encrypted index $\mathcal{I}$ and a search token τ_q as input. For each encrypted index entry C_i in $\mathcal{I}$ it checks if a value is indexed that is equal to the search token. That is, it checks if there exists a value $\gamma' \in C_i$ that is equal to τ_q, i.e. $\gamma' = \tau_q$. If this is the case, it adds the corresponding document identifier i to the result list $D[q]$. After iterating over all index entries $C_1, \ldots, C_d$, the algorithm outputs the result set $D[q]$.

Tasks

1. Describe in your own words the general approach how to prove an upper bound for the information leaked by an SSE scheme. Use this description as an outline for the following questions which are specific for the construction described above. *(2 Points)*

2. State the leakage you identified for the SSE scheme described in this assignment, try to restrict this leakage as much as possible. Give a brief explanation for each component of the leakage function you identified. *(4 Points)*

3. Sketch the security proof for the leakage you identified in the previous question against a non-adaptive adversary in the simulation-based framework under the assumption F is a secure PRF. Follow the approach you described in Task 1. *(4 Points)*

2 Attacking Property-preserving Encryption *(10 Points)*

Download the encrypted sqlite[1] database from Canvas that contains a list of (fictional) students with their first and last name together with their scores. As this is confidential

[1] `https://www.sqlite.org/index.html`

information, it is encrypted following the CryptDB [2] approach. Specifically, strings are encrypted deterministically[2] while grades are encrypted with order-preserving encryption. You know that the database contains information about students from all over the world, and you have a list of common first names (see `firstnames.txt`) and last names (see `lastnames.txt`) as background information. The lists are already sorted by frequency according to the appearance of corresponding names; not all names in the textfiles might be represented in the database. More specifically, if k unique last names are represented in the database, then you can assume these names are the first k names in `lastnames.txt` and similar for first names. Further, you know that the scores can range from 1 up to 100. Your task is to exploit the given background information and the weaknesses of property-preserving encryption.

Tasks

Answer the following questions and briefly explain your approach how you got to your given answer.

1. How many unique first names are stored in the encrypted database? *(1 Points)*
2. Plot the distribution of last names and analyse the graph. *(2 Points)*
3. What are the full names of all students who scored 99 points? *(3 Points)*
4. In the lecture we discussed an (α, t)-secure index as mitigation for attacks on searchable symmetric encryption as introduced by Islam et a. [4]. Implement this mitigation with your own choice of α and t for the last names.

   Plot the distribution of encrypted names after this mitigation has been applied. Explain the advantages and disadvantages of this approach with your chosen parameters. *(4 Points)*

References

[1] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. *Journal of Computer Security*, 19(5):895–934, 2011.

[2] Raluca Ada Popa, Catherine MS Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In *Proceedings of the twenty-third ACM symposium on operating systems principles*, 2011.

[3] Warren He, Devdatta Akhawe, Sumeet Jain, Elaine Shi, and Dawn Song. Shadowcrypt: Encrypted web applications for everyone. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.

[4] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *Proceedings of the Network and Distributed System Security Symposium*, 2012.

[2] More specifically, we used HMACs for indexable deterministic values as proposed in ShadowCrypt [3]