Improving precision of oblivious cross tags based searchable symmetric encryption scheme

I. Abstract

A Dynamic SSE (Searchable symmetric encryption) scheme should support updates and keyword searches in tandem to outsourced symmetrically encrypted data, while minimizing the amount of data revealed to the untrusted server. For this purpose, literature in this field has identified two critical security properties, i.e. forward and backward privacy. Forward privacy makes it hard for the server to correlate an update operation with previously executed search operations. Backward privacy limits the amount of information learnt by the server about documents that have already been deleted from the database.

In this work we develop upon the previously existing "Oblivious Cross Dynamic Tags" or ODXT scheme for conjunctive keyword searches. The goal is to improve the precision of given scheme and perform leakage analysis of the improved scheme.

II. Introduction

The ODXT scheme proposed in [] is first dynamic SSE scheme that is both forward and backward private. Its performance scales to very large arbitrarily-structured database, which includes both attribute-value and free-text databases.

Previously handling conjunctive searches also made the leakage analysis much more complicated, due the fact that based on previously existing schemes, which were tuned to handle leakage profiles of single keyword based searches, are significantly lacking for handling of conjunctive keyword based searches.

The oblivious cross dynamic tags scheme uses cross tag pairs without explicitly recovering the database, as done in "Basic dynamic cross tags" scheme. As suggested by the scheme name itself, it uses cross tags, oblivious to the untrusted server, which are generated dynamically to locate the presence of particular keyword tagged to a certain file without revealing the information about the keyword and the respective file for that particular update operation. Thus by dynamically computing the tags for keywords, other than *s-term*, on the untrusted server without revealing the information pertaining to them, we can effectively perform conjunctive keyword search.

III. Our Contribution

The Literature pertaing to the above said scheme, i.e. ODXT scheme, defines the given scheme but does not talk about the precision of the scheme. During the discourse about this scheme, we found that the oblivious cross tags generated from the *s-term*, generally fail consider the conjugate operation based tags for rest of the query terms. This results in an output which has 100% recall, disregarding its precision. This problem is well explained in next section *Improving precision.*

We attempted to rectify this problem by generating a *pair-factor,* β, which is used in similar fashion the ODXT scheme used *blinding-factor,* α.

IV. Improving precision

The ODXT scheme generates tags based on the oblivious, to untrusted server, information about the *s-term.* These tags pertain information regarding the operation type, which was performed for *s-term*, and the file id on which the operation is performed. Generally, these tags will generate following types of update sequence for a particular keyword,

1. *DEL-ADD*, where addition operation is performed after delete operation, with the same keyword on the same file id.

2. *ADD-DEL*, where delete operation is performed after addition operation, with the same keyword on the same file id.

As the ODXT scheme only computed tags based on the update operation sequence of the *s-term*, it failed to consider the above mentioned update operation sequence for remaining keywords or the query. Thus resulting in output which showcased only the *'add'* operation performed for keywords on resultant files, since *'del'* for *s-term* implies exclusion of the file id from result, thus complying with the premise of conjunctive keyword search.

To solve the problem of considering *ADD-DEL* update sequence of keywords, excluding *s-term*, we need to create *xtag* for the conjugate operation along with the order in which they are performed.

Dynamic Pair-factor in TSet. The client also computes and stores in the TSet dictionary a dynamic pair element corresponding to each update operation This *pair-factor,* β, to look up for the conjugate operations, can be defined as following,

$$\beta = (F_p(K_y, id_j||op))^{-1} F_p(K_y, id_j||op^{'}) \tag{1}$$

where F_p is a pseudo-random function, with value $\in Z_p^*$, K_y is the key used by F_p.

By storing this value along-side *blinding-factor,* α, we can compute, tags with conjugate operation.

$$\alpha = F_p(K_y, id_j||op)(F_p(K_z, w_{s-term}||cnt)^{-1})$$

$$\alpha\beta = F_p(K_y, id_j||op^{'})(F_p(K_z, w_{s-term}||cnt)^{-1})$$

To keep track of the order in which the pair of update operation is performed the server keeps sequence of all update operation, This is performed by simply saving the update operation time-stamp or id in XSet.

$$XSet[xtag] = update_{id/time}$$

By utilising these values the order can be inferred resulting in identifying the possibility of *ADD-DEL* sequence, which can be used to further reduce the result of scheme to only those file ids which comply with the conjunctive keyword search.

V. Analysis

Computational complexities: Since the algorithm of the original scheme is modified very little, this modified algorithm has same computational complexities as that of original, ODXT scheme, algorithm.

Leakage Analysis: Due to the modification the untrusted server will be able to gain knowledge of sequence of operation performed and will be able to retain the knowledge of execution of conjugate operation as well, while begin unable to determine which operation was performed. Thus, when an update operation is performed, the untrusted server will only be able to infer if its conjugated operation is also performed or not