1 Papers referred to

1. Raphael Bost and Pierre-Alain Fouque. **Thwarting Leakage Abuse Attacks against Searchable Encryption - A Formal Approach and Applications to Database Padding**

2 Discussion of the reference paper

- Their security model adds new constraints the standard security model (as of ESE). The constraints are rather arbitrary, so there is no universal security guarantees.
- Their construction against count attack relies on padding sets of keywords to the same frequency, but that is vulnerable to variance attack.
- If the fake documents are all single-keyword ones, the adversary can simply remove all those documents first and then run count attack. Although some information is lost (on the real single-keyword documents), the recovery attack is as efficient as the unpadded version.
- Their scheme is not secure in experiments, and not secure provably.

3 New Padding Scheme

3.1 The idea

- Full padding is very costly. On the experiment database, full padding after merging alphabets in attribute columns requires an expansion factor of over 17000, so it is completely not practical.
- If all we require is that the alphabets in attribute columns are indistinguishable (we are willing to leak the distribution of attributes), padding can be done much more efficiently. See dedicated document.
- Padding fake alphabets to the columns are ok, as long as they are queried every now and then.

3.2 Main challenge

- We require size of alphabets for each column to be either the same or co-prime (except the largest alphabet size), that means it requires full padding between those two attributes.
- If all alphabet sizes are multiples of each other, it is easy to achieve indistinguishability for pairs of attributes. But attribute column 0 and attribute column 1 are indistinguishable, and attribute column 0 and attribute column 2 are indistinguishable, does not imply attribute column 1 and attribute 2 are indistinguishable using simple counting argument.
- **New idea:** (1) Fix a permutation on unit block. (2) Permute set of assignments on unit blocks to generate new rows. (3) Repeat the large block. Note: this requires

4 Security Notions

4.1 Adversary for count attack

- We want to define formally what is an adversary for count attack with respect to the syntax(s) of databases.

4.2 Leakage abuse attack

- What does leakage really mean?
- Idea: we can think of leakage as a map. e.g. In count attack, for keyword set $\mathcal{W}$, leakage can be modelled as a function $\mathcal{L} : \mathcal{W} \times \mathcal{W} \to \mathbb{N}$ so $\mathcal{L}(w_0, w_1)$ is mapped to the co-occurrence count of keywords w_0 and w_1. Count attack works exactly because a large proportion of co-occurrence counts are unique, so $\mathcal{L}$ is invertible on the corresponding restricted domain. The restricted domain can be written down quite nicely:

  $$\{(v, w) \mid \ (\mathcal{L}(v, w) \neq \mathcal{L}(v', w') \text{ for any } v', w' \text{ such that } v' \prec w') \text{ and } (v \prec w)\}$$

  where $(\mathcal{W}, \prec)$ is a well order on keyword set $\mathcal{W}$.
- Consider our padding scheme where we require permutations of keywords to have the same count. Obviously this leakage function does not apply because we are not leaking the exact counts. But let's see what we get if we use this leakage function on the padded database.

  Using the example: (0,0) = 15, (0,1) = 13, (1,0) = 13, (1,1) = 15. We see $\mathcal{L}^{-1}(15) = \{(0, 0), (1, 1)\}$ but this information is not helpful in identifying