A Highly Accurate Query-Recovery Attack against Searchable Encryption using Non-Indexed Documents

Marc Damie*, Florian Hahn, Andreas Peter

August 11, 2021
Motivations
Searchable Symmetric Encryption (SSE)
Related works

- Scope: Passive query-recovery attacks against SSE
- SSE schemes leak the access pattern and the search pattern
- All these attacks exploit this leakage to compute a trapdoor-trapdoor co-occurrence and compare it to a keyword-keyword co-occurrence obtained using documents known by the attacker
- Known-data attacks (when attacker-known documents are indexed) vs. Similar-data attacks (when the documents are only similar, i.e. non-indexed)
Previous attacks


- Cash et al. (2015): Based on a filtering approach. Significantly better than Islam et al.’s attack but still only effective as a known-data attack.


- Blackstone et al. (2020): Based on a filtering approach. By construction, can only be used as a known-data attack. Reduce drastically the amount of known documents needed compared to the previous attacks.

- **Summary**: no effective/accurate similar-data attack. Known-data setup can be considered as a strong (unrealistic?) assumption.
Other types of attacks

- Attack with a malicious attacker: Zhang et al. (2016)
- Attack on schemes supporting range queries: Kellaris et al. (2016), Grubbs et al. (2018), Lacharité et al. (2018)
- Other types of attacks exist but are out of scope because they assume a different type of attacker knowledge, a different threat model, a different search scheme, etc.
Our contributions

• A scoring approach to design effective attacks with interpretable results

• Weakening of the attacker assumptions by proposing a highly effective similar-data attack achieving recovery rates of up to 90%

• A proper formalization of the concept of similarity for document sets

• Extensive analysis of our best attack: its qualities and its limitations
Attacker knowledge

- Similar document set: documents similar but different to the indexed documents ⇒ extract a vocabulary and a word-word co-occurrence matrix

- Observed queries: the attacker has observed some queries ⇒ compute a trapdoor-trapdoor co-occurrence matrix

- Known queries: for a small part of the observed queries, knows the underlying keyword
Creating a keyword/trapdoor vector

Known queries = [(Koala, ), ... (Shark, )]
+ keyword-keyword co-occurrence matrix
+ trapdoor-trapdoor co-occurrence matrix

\[
\text{Vect(Cat)} = [\text{Coocc(Cat, Koala)}, \ldots \text{Coocc(Cat, Shark)}]
\]

\[
\text{Vect( )} = [\text{Coocc( , )}, \ldots \text{Coocc( , )}]
\]

Figure: Attacker knowledge transformation
Scoring function

$$\text{MatchingScore(Cat, } \mathbf{\hat{v}} \text{) = - \ln(\|\text{Vect(Cat)} - \text{Vect(} \mathbf{\hat{v}} \text{)}\|)}$$

- Using this vectorization, we can directly compare trapdoors to keywords
- The matching score is a logarithmic transformation of a distance between a keyword vector and a trapdoor vector
- Having a score provides a result interpretability: the higher a score is, the more likely a given prediction is
Attack algorithm

- Compute the matching score of each trapdoor-keyword pair and return the keyword providing the highest score for each trapdoor.

- Very fast (few seconds) and deterministic.

- Exploitable prediction scores. Can be used to design improvement strategies (e.g. refinement and clustering presented in the paper).
Experimental setup

- Each result is the average accuracy over 50 experiments
- The indexed document set and the attacker document set are two randomly picked disjoint subsets of the Enron document set
- The attacker does not know the queryable vocabulary contrary to the previous attack papers
- The vocabulary is the $m$ most frequent keywords of the indexed document set. By default, we use $m = 1K$
- The queries are uniformly picked among the queryable vocabulary. By default, the query set size is 15% of the vocabulary size
- In the paper, we test different sizes for the vocabulary, the query set, etc
Experimental results

*Comment:* improves the state-of-the-art but still impractical (no. of known queries needed too high).
Refined score attack
Refinement strategy

**Goal**: reduce drastically the number of known queries needed.

We iteratively impute new known queries. Three steps per iteration:

1. Remove all (attacker-)known queries from the queries to be recovered
2. Use the base score attack to find a candidate for each unknown query/trapdoor. Use the score to evaluate each prediction "certainty"
3. If there are more than $k$ remaining unknown queries, add the $k$ most certain queries to the known query set. Otherwise, stop the algorithm and return the predictions
Experimental results

**Figure:** Score attack vs. Refined score attack
We propose a similarity metric $\epsilon$ to compare document sets. The attacker assumes that $D_{\text{real}}$ and $D_{\text{sim}}$ are $\epsilon$-similar, with $\epsilon$ sufficiently small.
Refined attack mitigation

**Figure:** Comparison of the accuracy for two countermeasures.
Conclusion

- Highly accurate attacks using non-indexed documents are possible (Score and Refined Score attacks being two examples)

- Our attacks work under weaker assumptions on the attacker’s background knowledge than previously published attacks and move toward realistic and practical attack situations

- Despite the accuracy of the Refined Score attack, even the simplest countermeasures can be effective (at the cost of some overheads)
Thank you for your attention!

**Code available:** https://github.com/MarcT0K/Refined-score-atk-SSE

Feel free to contact us:

→ marc.damie@etu.utc.fr
→ f.w.hahn@utwente.nl
→ a.peter@utwente.nl