TWo-IN-one-SSE: Fast, Scalable and Storage-Efficient Searchable Symmetric Encryption for Conjunctive and Disjunctive Boolean Queries - Supplemental Material

We state the adaptive security game for SSE, the proofs for Theorem 4.2 and Theorem B.1 (in supplemental material) in this section.

A Adaptive Security of SSE

The adaptive security of any SSE scheme is parameterized by a leakage function

$$\mathcal{L} = \left(\mathcal{L}^{\textsc{Setup}}, \mathcal{L}^{\textsc{Search}}\right),$$

where $\mathcal{L}^{\textsc{Setup}}$ encapsulates the leakage to an adversarial server during the setup phase, and $\mathcal{L}^{\textsc{Search}}$ encapsulates the leakage to an adversarial server during each execution of the search protocol.

Algorithm 1 Experiment $\mathbf{Real}^{\mathsf{SSE}}(\lambda, Q)$

```
function $\mathbf{Real}^{\mathsf{SSE}}(\lambda, Q)$
    $N \leftarrow \mathbf{Adv}(\lambda)$
    $(\mathsf{sk}, \mathsf{st}_0, \mathbf{EDB}_0) \leftarrow \textsc{Setup}(\lambda, N)$
    for $k \leftarrow 1 \text{to} Q$ do
        Let $q_k \leftarrow \mathbf{Adv}(\lambda, \mathbf{EDB}_{k-1}, \tau_1, \ldots, \tau_{k-1})$
        Let $(\mathsf{st}_k, \mathbf{EDB}_k, \mathbf{DB}(q_k)) \leftarrow$
            $\textsc{Search}(\mathsf{sk}, \mathsf{st}_{k-1}, q_k; \mathbf{EDB}_{k-1})$
        Let $\tau_k$ denote the view of the adversary after
            the $k^{\text{th}}$ query
    $b \leftarrow \mathbf{Adv}(\lambda, \mathbf{EDB}_Q, \tau_1, \ldots, \tau_Q)$
    return $b$
```

Informally, an SSE scheme is adaptively secure with respect to a leakage function $\mathcal{L}$ if the adversarial server provably learns no more information about **DB** other than that encapsulated by $\mathcal{L}$. Formally, an SSE scheme is said to be adaptively secure with respect to a leakage function $\mathcal{L}$ if for any stateful PPT adversary

Algorithm 2 Experiment $\mathbf{Ideal}^{\mathsf{SSE}}(\lambda, Q, \mathcal{L})$

```
function $\mathbf{Ideal}^{\mathsf{SSE}}(\lambda, Q, \mathcal{L})$
    Parse the leakage function $\mathcal{L}$ as:
    $\mathcal{L} = \left(\mathcal{L}^{\textsc{Setup}}, \mathcal{L}^{\textsc{Search}}\right)$.
    $(\mathsf{st}_{\mathrm{SIM}}, \mathbf{EDB}_0) \leftarrow \textsc{SimSetup}(\mathcal{L}^{\textsc{Setup}}(\lambda, N))$
    for $k \leftarrow 1$ to $Q$ do
        Let $q_k \leftarrow \mathbf{Adv}(\lambda, \mathbf{EDB}_{k-1}, \tau_1, \ldots, \tau_{k-1})$
        Let $(\mathsf{st}_{\mathrm{SIM}}, \mathbf{EDB}_k, \tau_k) \leftarrow \textsc{SimSearch}$
            $(\mathsf{st}_{\mathrm{SIM}}, \mathcal{L}^{\textsc{Search}}(q_k); \mathbf{EDB}_{k-1})$
        Let $\tau_k$ denote the view of the adversary after
            the $k^{\text{th}}$ query
    $b \leftarrow \mathbf{Adv}(\lambda, \mathbf{EDB}_Q, \tau_1, \ldots, \tau_Q)$
    return $b$
```

$\mathbf{Adv}$ that issues a maximum of $Q = \mathsf{poly}(\lambda)$ queries, there exists a stateful probabilistic polynomial-time simulator $\textsc{Sim} = (\textsc{SimSetup}, \textsc{SimSearch})$ such that the following holds:

$$\left|\Pr\left[\mathbf{Real}^{\mathsf{SSE}}_{\mathbf{Adv}}(\lambda, Q) = 1\right] - \Pr\left[\mathbf{Ideal}^{\mathsf{SSE}}_{\mathbf{Adv},\mathrm{SIM}}(\lambda, Q) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

where the "real" experiment $\mathbf{Real}^{\mathsf{SSE}}$ and the "ideal" experiment $\mathbf{Ideal}^{\mathsf{SSE}}$ are as described in Algorithm 1 and Algorithm 2 (in Appendix).

B Security of $\text{TWINSSE}_{\text{OXT}}$

We now formalize the security of $\text{TWINSSE}_{\text{OXT}}$ in terms of the leakage profiles described above. We do this using a formal theorem, which may be viewed as a specialization of Theorem 4.2 (in the main paper) to a specific instantiation of TWINSSE based on OXT. Once again, this theorem is based on the (adaptive) simulation-security definition of SSE in the real world-ideal world paradigm, described in Appendix A.

Theorem B.1 (Security of $\text{TWINSSE}_{\text{OXT}}$)**.** $\text{TWINSSE}_{\text{OXT}}$ *is an (adaptively) secure SSE scheme with respect to the leakage function* $\mathcal{L}_{\text{TWINSSE}_{\text{OXT}}} = (\mathcal{L}^{\textsc{Setup}}_{\text{TWINSSE}_{\text{OXT}}}, \mathcal{L}^{\textsc{Search}}_{\text{TWINSSE}_{\text{OXT}}})$, *where for any plaintext database* $\mathbf{DB}$, *any sequence of conjunctive queries* $\mathcal{Q}_0 = (q_{1,0}, \ldots, q_{M,0})$ *and any sequence of disjunctive queries* $\mathcal{Q}_1 = (q_{1,1}, \ldots, q_{M',1})$, *and any pair of bucketization parameters* (n', n_{B}), *we have*

$$\mathcal{L}^{\textsc{Setup}}_{\text{TWINSSE}_{\text{OXT}}}(\mathbf{DB}) = (|\widehat{\mathbf{DB}}|, n', n_{\mathrm{B}}),$$

where $\widehat{\mathbf{DB}} = \textit{GenMetaDB}(\mathbf{DB}, n', n_{\mathrm{B}})$*, and*

$$\mathcal{L}^{\textsc{Search}}_{\text{TWINSSE}_{\text{OXT}}}(\mathcal{Q}_0, \mathcal{Q}_1) = [\mathsf{RP}, \mathsf{SP}, \mathsf{EP}, \mathsf{IP}](\mathcal{Q}_0, \mathcal{Q}_{\mathsf{mkw},1}),$$

where $\mathcal{Q}_{\mathsf{mkw},1}$ is a sequence of (sub-)meta-queries of the form

$$\mathcal{Q}_{\mathsf{mkw},1} = \{q_{\mathsf{mkw},k,\ell}\}_{k\in[n_{\mathsf{B}}],\ell\in[M']},$$

where for each $\ell \in [M']$, we have

$$q_{\mathsf{mkw},\ell} = \left(\bigvee_{k\in[n_{\mathsf{B}}]} q_{\mathsf{mkw},k,\ell}\right) = \textit{GenMQuery}(q_{\ell,1}, n', n_{\mathsf{B}}).$$

C Proof of Theorem 4.2 (Security Analysis of TWINSSE)

We provide a simulation-based proof approach for TWINSSE. We assumed that the underlying adaptively secure CSSE has the following leakage profile.

$$\mathcal{L}_{\text{CSSE}} = (\mathcal{L}^{\textsc{Setup}}_{\text{CSSE}}, \mathcal{L}^{\textsc{Search}}_{\text{CSSE}})$$

We express the leakage of TWINSSE as,

$$\mathcal{L}_{\text{TWINSSE}} = (\mathcal{L}^{\textsc{Setup}}_{\text{TWINSSE}}, \mathcal{L}^{\textsc{Search}}_{\text{TWINSSE}})$$

where,

$$\mathcal{L}^{\textsc{Setup}}_{\text{TWINSSE}}(\mathbf{DB}) = \mathcal{L}^{\textsc{Setup}}_{\text{CSSE}}(\widehat{\mathbf{DB}})$$

and, $\widehat{\mathbf{DB}} = \textsc{GenMeta}\text{DB}(\mathbf{DB}, n', n_{\mathsf{B}})$, and

$$\mathcal{L}^{\textsc{Search}}_{\text{TWINSSE}}(q) = \begin{cases} \mathcal{L}^{\textsc{Search}}_{\text{CSSE}}(q) & \text{if q is conjunctive,} \\ \{\mathcal{L}^{\textsc{Search}}_{\text{CSSE}}(q_{\mathsf{mkw},k})\}_{k\in[n_{\mathsf{B}}]} & \text{if q is disjunctive,} \end{cases}$$

where

$$q_{\mathsf{mkw}} = \left(\bigvee_{k\in[n_{\mathsf{B}}]} q_{\mathsf{mkw},k}\right) = \textsc{GenMQuery}(q, n', n_{\mathsf{B}}).$$

We show that TWINSSE is secure against an adaptive semi-honest adversary $\mathcal{A}$, which has access to leakages from TWINSSE. We build a simulator SIM $\widehat{\mathbf{EDB}}$ generation by TWINSSE.Setup, and transcripts for queries over $\widehat{\mathbf{EDB}}$. The simulator simulates the transcripts τ_i for each query q_i. The simulator has the inputs from the leakage function $\mathcal{L}_{\text{TWINSSE}}$ only, with the setup leakage $\mathcal{L}^{\textsc{Setup}}_{\text{TWINSSE}}$ and the search leakage $\mathcal{L}^{\textsc{Search}}_{\text{TWINSSE}}$.

Simulating TWINSSE.Setup**:** The following public parameters are available to SIM_{CSSE} as a part of $\mathcal{L}^{\textsc{Setup}}_{\text{TWINSSE}}$.

$$\{\mathbf{DB}, n', n_{\mathsf{B}}\}$$

The simulator outputs the its version of $\widehat{\textbf{EDB}}$ according to the simulation process of CSSE (we assumed that CSSE is provably simulation secure).

$$\begin{aligned} ct_{\widehat{\textbf{EDB}}} &= SIM^{\textsc{Setup}}_{\text{TWINSSE}}(\textbf{DB}) \\ &= SIM^{\textsc{Setup}}_{\text{CSSE}}(\widehat{\textbf{DB}}) \\ &= SIM^{\textsc{Setup}}_{\text{CSSE}}(\textbf{DB}, n', n_B) \end{aligned}$$

Since, CSSE is proven simulation secure, if follows from the simulation security guarantee of CSSE that $ct_{\widehat{\textbf{EDB}}}$ is indistinguishable from the one generated in the real experiment.

Simulating TWINSSE.Search**:** For conjunctive queries the adversary does not have any advantage from $\mathcal{L}^{\textsc{Search}}_{\text{TWINSSE}}$ compared to $\mathcal{L}^{\textsc{Search}}_{\text{CSSE}}$, which exactly same as CSSE. For disjunctive queries we consider the effect of querying using q_{mkw}.

For disjunctive queries, we argue that the adversary $\mathcal{A}$ does not gain any information about the original disjunctive query with this simulation experiment. The distribution of $\widehat{\textbf{DB}}$ (hence, also for $\widehat{\textbf{EDB}}$) is abstracted from **DB** by the meta-keywords. The search leakages of CSSE is characterised by the $\mathcal{L}_{\text{CSSE}}$, provided from CSSE construction. Since, CSSE in TWINSSE executes over meta-keyword only, this leakage is expressed in the context of meta-keywords as below.

$$\mathcal{L}'_{\text{CSSE}} = \mathcal{L}_{\text{CSSE}}(meta - keywords)$$

With this leakage information of CSSE, the search leakage of TWINSSE can be expressed as below.

$$\mathcal{L}^{\textsc{Search}}_{\text{TWINSSE}}(q) = \mathcal{L}^{\textsc{Search}}_{\text{TWINSSE}}(q_{\mathsf{mkw},k})_{k\in[n_{\text{B}}]} = \{\mathcal{L}'_{\text{CSSE}}, n_{\text{B}}, n'\}$$

The parameters n_{B} and n' are derived from N (number of keywords), which is available during setup. Therefore, the search leakage of TWINSSE same as the underlying CSSE, which can be summarised as below.

$$\mathcal{L}^{\textsc{Search}}_{\text{TWINSSE}}(q) = \mathcal{L}^{\textsc{Search}}_{\text{TWINSSE}}(q_{\mathsf{mkw},k})_{k\in[n_{\text{B}}]} = \{\mathcal{L}'_{\text{CSSE}}\}$$

This same leakage profile for search in TWINSSE and CSSE in the context of meta-keywords ensures that no additional information is leaked beyond CSSE leakage.

D Proof of Theorem B.1 (Security Analysis of TWINSSE$_{\text{OXT}}$)

We resort to a simulation-based security analysis for TWINSSE$_{\text{OXT}}$. We assume a semi-honest adversary $\mathcal{A}$ which has access to the leakage from standard SSE leakages in an adaptive model. Security analysis of TWINSSE relies upon the semantic security notions provided by CSSE. TWINSSE inherits these notions through the core OXT (in case of TWINSSE$_{\text{OXT}}$, the OXT) instance. We assume the following properties of OXT achieves with efficient performance.

1. Primitives used in construction of OXT hold the standard security assumptions.

2. OXT is non-adaptively and adaptively secure with the above assumptions.

We consider the following leakage profile for OXT.

$$\mathcal{L}_{\mathsf{OXT}} = \{\mathcal{L}_{\mathsf{OXT}}^{\textsc{Setup}}, \mathcal{L}_{\mathsf{OXT}}^{\textsc{Search}}\}$$

Here, $\mathcal{L}_{\mathsf{OXT}}^{\textsc{Setup}}$ captures the leakage from the OXT.Setup, and $\mathcal{L}_{\mathsf{OXT}}^{\textsc{Search}}$ encapsulates the leakage from OXT.Search. More precisely, these can be expressed as,

$$\mathcal{L}_{\mathsf{OXT}}^{\textsc{Setup}}(\mathbf{DB}) = \{|\mathbf{DB}|\}$$

and

$$\mathcal{L}_{\mathsf{OXT}}^{\textsc{Search}}(\mathbf{EDB}, \{q_k\}_{q_k \in \mathcal{Q}_0)} = \{RP, SP, EP, IP\}$$

where, $\mathcal{Q}_0$ is a set of conjunctive queries. The leakages RP, SP, EP, and IP are the pattern leakages from OXT (see Appendix A of the main paper).

We define the leakage profile of $\text{TWINSSE}_{\mathsf{OXT}}$ with respect to these above definitions and assumptions as below.

$$\mathcal{L}_{\text{TWINSSE}_{\mathsf{OXT}}} = \{\mathcal{L}_{\text{TWINSSE}_{\mathsf{OXT}}}^{\textsc{Setup}}, \mathcal{L}_{\text{TWINSSE}_{\mathsf{OXT}}}^{\textsc{Search}}\}$$

The leakage functions above can be expressed as

$$\mathcal{L}_{\text{TWINSSE}_{\mathsf{OXT}}}^{\textsc{Setup}}(\mathbf{DB}) = \{|\widehat{\mathbf{DB}}|, n', n_{\mathsf{B}}\}$$

and

$$\mathcal{L}_{\text{TWINSSE}_{\mathsf{OXT}}}^{\textsc{Search}}(\widehat{\mathbf{EDB}}, \mathcal{Q}_0, \mathcal{Q}_1) = [\mathsf{RP}, \mathsf{SP}, \mathsf{EP}, \mathsf{IP}](\mathcal{Q}_0, \mathcal{Q}_{\mathsf{mkw},1}),$$

.

For conjunctive queries,

$$\mathcal{L}_{\text{TWINSSE}_{\mathsf{OXT}}}^{\textsc{Search}}(\widehat{\mathbf{EDB}}, \{q_k\}_{q_k \in \mathcal{Q}_0}) = [\widehat{\mathsf{RP}}, \widehat{\mathsf{SP}}, \widehat{\mathsf{EP}}, \widehat{\mathsf{IP}}]$$

.

Here, $\{\widehat{\mathsf{RP}}, \widehat{\mathsf{SP}}, \widehat{\mathsf{EP}}, \widehat{\mathsf{IP}}\}$ are the $\{\mathsf{RP}, \mathsf{SP}, \mathsf{EP}, \mathsf{IP}\}$ leakages in the context of meta-keywords. For conjunctive queries, it is exactly the same as OXT.

Since, OXT is simulation secure against these leakages, simulation security of $\text{TWINSSE}_{\mathsf{OXT}}$ for conjunctive queries is straightforwardly implied from OXT.

In disjunctive queries, the query transformation process is carried out locally by the client, and the actual search is completed using OXT.Search protocol, we can write $\text{TWINSSE}_{\mathsf{OXT}}$.Search leakage as

$$\mathcal{L}_{\text{TWINSSE}_{\mathsf{OXT}}}^{\textsc{Search}}(\widehat{\mathbf{EDB}}, \{q_{mkw,1,k}\}_{k \in [\mathcal{Q}_1]}) = \{\widehat{\mathsf{RP}}, \widehat{\mathsf{SP}}, \widehat{\mathsf{EP}}, \widehat{\mathsf{IP}}\}$$

.

We build a simulator SIM to simulate the $\widehat{\mathbf{EDB}}$ generation by TWINSSE$_{\mathsf{OXT}}$ from **DB**, and transcripts for query search over $\widehat{\mathbf{EDB}}$. The simulator simulates the transcripts τ_i for each query $q_i \in \mathcal{Q}$. The simulator has the inputs from the leakage function $\mathcal{L}_{\text{TWINSSE}_{\mathsf{OXT}}}$ only, with the setup leakage $\mathcal{L}^{\textsc{Setup}}_{\text{TWINSSE}_{\mathsf{OXT}}}$ and the search leakage $\mathcal{L}^{\textsc{Search}}_{\text{TWINSSE}_{\mathsf{OXT}}}$.

Simulating Setup: The following public parameters are available to SIM_{OXT} as a part of $\mathcal{L}^{\textsc{Setup}}_{\text{TWINSSE}_{\mathsf{OXT}}}$.

$$\{|\mathbf{EDB}|, |\widehat{\Delta}|\}$$

The simulator outputs the its version of $\widehat{\mathbf{EDB}}$ according to the simulation process of OXT (we assumed that OXT is provably simulation secure).

$$ct_{\widehat{\mathbf{EDB}}} = SIM_{\mathsf{OXT}}.\textsc{Setup}(|\mathbf{MDB}|, |\widehat{\Delta}|)$$

It follows from the simulation security guarantee of OXT that $\mathsf{ct}_{\widehat{\mathbf{EDB}}}$ is indistinguishable from the one generated in the real experiment.

Simulating Search: For the conjunctive queries, the leakage $\mathcal{L}^{\textsc{Search}}_{\text{TWINSSE}_{\mathsf{OXT}}}$ is exactly the same as $\mathcal{L}^{\textsc{Search}}_{\text{OXT}}$. Hence, we can write the following.

$$\mathcal{L}^{\textsc{Search}}_{\text{TWINSSE}_{\mathsf{OXT}}}(\widehat{\mathbf{EDB}}, \{q_k\}_{k\in[|\mathcal{Q}|]}) = \mathcal{L}^{\textsc{Search}}_{\text{OXT}}(\mathbf{EDB}, \{q_k\}_{k\in[|\mathcal{Q}|]})$$

By the simulation security guarantee of OXT, TWINSSE$_{\mathsf{OXT}}$ secure against these leakages.

For disjunctive queries, we argue that the adversary $\mathcal{A}$ does not gain any information about the original disjunctive query except $|q|$. The distribution of **MDB** (encrypted to $\widehat{\mathbf{EDB}}$) is abstracted from **DB** through the meta-keywords. We resort to a more conservative analysis for this proof, as keywords do not have direct inference from meta-keywords, especially that is applicable over any database in general. The position of each w in an mkw is fixed according to the frequency of w, which is unique for a **DB**. The lemmas below relate worst cases where an inference can be established between the query keywords and the corresponding meta-keywords without any additional knowledge of the plain database.

Lemma D.1, Lemma D.2, and Lemma D.3 relates the disjunctive q with $\mathsf{w}_i \in \Delta$ to the conjunctive q with $\mathsf{mkw}_i \in \widehat{\Delta}$.

Lemma D.1. *Consider two disjunctive queries of the same length t*

$$q_0 = \mathsf{w}_{1,q_0} \vee \mathsf{w}_{2,q_0} \vee \ldots \vee \mathsf{w}_{t,q_0},\ \mathsf{w}_{i,q_0}$$
$$q_1 = \mathsf{w}_{1,q_1} \vee \mathsf{w}_{2,q_1} \vee \ldots \vee \mathsf{w}_{t,q_1},\ \mathsf{w}_{i,q_1}$$

have the following expressions using mkw*s,*

$$q_0 = q_{0,\mathsf{mkw}} = \mathsf{mkw}_{1,q_0} \wedge \mathsf{mkw}_{2,q_0} \wedge \ldots \wedge \mathsf{mkw}_{t+1,q_0}$$
$$q_1 = q_{1,\mathsf{mkw}} = \mathsf{mkw}_{1,q_1} \wedge \mathsf{mkw}_{2,q_1} \wedge \ldots \wedge \mathsf{mkw}_{t+1,q_1}$$

both of length $t+1$, and the mkw*s are placed in the increasing order of the starting index of the* 0*s stretch in each* mkw*. If the* mkw*s at index k in q_0 and q_1 are the same, then* $\mathsf{w}_{k-1,q_0} = \mathsf{w}_{k-1,q_1}$ *and* $\mathsf{w}_{k,q_0} = \mathsf{w}_{k,q_1}$.

Proof. The proof of Lemma D.1 is given in Section D.1.1. □

Lemma D.2. *Consider two disjunctive queries q_0 and q_1, of the same length t have the* mkw *expressions as defined in Lemma D.1 - both of length $t+1$. If the* mkw*s at indices k_0 in q_0, and k_1 in q_1 are the same, then* $\mathbf{x}_{k_0-1,q_0} = \mathsf{w}_{k_1-1,q_1}$ *and* $\mathsf{w}_{k_0,q_0} = \mathsf{w}_{k_1,q_1}$.

Proof. The proof of Lemma D.2 is given in Section D.1.2. □

Lemma D.3. *Consider two disjunctive queries of different length t_0 and t_1 -*

$$q_0 = \mathsf{w}_{1,q_0} \vee \mathsf{w}_{2,q_0} \vee \ldots \vee \mathsf{w}_{t_0,q_0},\ \mathsf{w}_{i,q_0} \in \Delta$$
$$q_1 = \mathsf{w}_{1,q_1} \vee \mathsf{w}_{2,q_1} \vee \ldots \vee \mathsf{w}_{t_1,q_1},\ \mathsf{w}_{i,q_1} \in \Delta$$

have following expressions in the mkw*s*

$$q_0 = q_{0,\mathsf{mkw}} = \mathsf{mkw}_{1,q_0} \wedge \mathsf{mkw}_{2,q_0} \wedge \ldots \wedge \mathsf{mkw}_{t_0+1,q_0}$$
$$q_1 = q_{1,\mathsf{mkw}} = \mathsf{mkw}_{1,q_1} \wedge \mathsf{mkw}_{2,q_1} \wedge \ldots \wedge \mathsf{mkw}_{t_1+1,q_1}$$

which are of lengths t_0+1 and t_1+1 respectively. If the mkw*s at indices k_0 in q_0, and k_1 in q_1 are the same, then* $\mathsf{w}_{k_0-1,q_0} = \mathsf{w}_{k_1-1,q_1}$ *and* $\mathsf{w}_{k_0,q_0} = \mathsf{w}_{k_1,q_1}$.

Proof. The proof of Lemma D.3 is given in Section D.1.3. □

Recall that, the query transformation is executed by the client locally. The search is executed as a two-party protocol between the client and the server using the meta-keywords. The server learns $|q|$ trivially from q_{mkw} through of meta-keywords. From Lemma D.1, D.2, and D.3, an adversary can infer the position of the same ws in two queries of same length or different lengths if both queries have a *common* mkw in them.

However, the server can only infer if the least-frequent mkws in q_{mkw} are identical or not in mkw expressions of two qs from $\widehat{SP}$. The mkw expressions in each of the three lemmas require to place mkws in increasing order of the starting index of the 0's stretch. Whereas, the actual query expression for OXT has the least-frequent mkw first. No direct inference can be conjectured for the least-frequent mkw and the query expressions in the lemmas. Hence, an adversary $\mathcal{A}$ can not distinguish between the common meta-keyword and a distinct meta-keyword.

In the case, where the least-frequent of mkws is the first one in the query expression of the lemmas too, the first keyword is also the same for both ws. This is equivalent to the case of two conjunctive queries in keywords having the least-frequent w same.

Therefore, the leakage from TWINSSE$_{\text{SEARCH}}$ can be limited to the OXT pattern leakages only, as expressed below.

$$\mathcal{L}^{\text{SEARCH}}_{\text{TWINSSE}_{\text{OXT}}}(\widehat{\mathbf{EDB}}, \{q_k\}_{k\in[|\mathcal{Q}|]}) = \{\mathcal{L}'_{\text{OXT}}, |q_k|_{k\in[|\mathcal{Q}|]}\}$$

Since, OXT is proven simulation secure, if follows from the simulation security guarantee that $\mathcal{A}$ no additional advantage over the real experiment.

D.1 Proofs of the Lemmas

We present the proofs of the lemmas presented earlier in this section. We follow the notations and conventions as used in the main body of the paper.

D.1.1 Proof of Lemma D.1

Proof. By construction, each meta-keyword mkw_i has the original keywords appearing in sorted order in the binary string representation (increasing order of frequency from left to right). Assume, the k'th meta-keyword mkw_k is same for both the queries q_0 and q_1. Without loss of generality, a meta-keyword in the basic $O(N^2)$ (TWINSSE$_{\text{BASIC}}$) method can be formed as

$$\{b_1, b_2, \ldots, b_r, b_{r+1}, \ldots, b_s, b_{s+1}, \ldots, b_n\},\ b_i \in \{0, 1\}$$

where $1 \le r < s \le n$, and $b_i = 0$ for $r < i < s$.

To have an mkw of this form, q must have two keywords at indices r and s, and none in between (for q_0 and q_1 both). Since the mkws are constructed using ws in sorted order, if both queries q_0 and q_1 have the same r and same s (as one mkw is the same), the keywords w_r and w_s in both q_0 and q_1 are also the same. Hence, we have $\mathsf{w}_{k-1,q_0} = \mathsf{w}_{k-1,q_1}$ and $\mathsf{w}_{k,q_0} = \mathsf{w}_{k,q_1}$. □

D.1.2 Proof of Lemma D.2

Proof. We assume the common mkw of q_0 and q_1 can be expressed as

$$\{b_1, b_2, \ldots, b_r, b_{r+1}, \ldots, b_s, b_{s+1}, \ldots, b_n\},\ b_i \in \{0, 1\}$$

where $1 \le r < s \le n$, and $b_i = 0$ for $r < i < s$. The mkw appears at indices k_0 in q_0 and at k_1 in q_1. Since the indices of ws in the mkw strings are in sorted order (increasing frequency) and remains fixed for all mkws, the ws at index r and index s are the same for both q_0 and q_1. However, as the index of mkw is different in q_0 and q_1, the number of preceding ws before index r in q_0 and q_1 are different, equal to $k_0 - 2$ and $k_1 - 2$ respectively. Hence, for q_0, r is equal to $k_0 - 1$, and equal to $k_1 - 1$ in q_1. Following the above argument, we have $\mathsf{w}_{k_0-1,q_0} = \mathsf{w}_{k_1-1,q_1}$ and $\mathsf{w}_{k_0,q_0} = \mathsf{w}_{k_1,q_1}$. □

D.1.3 Proof of Lemma D.3

Proof. The proof of Lemma D.3 follows from the proof of Lemma D.2. Essentially, Lemma D.3 is the extension of Lemma D.2 for two different lengths of queries. Intuitively, it can be established in the following way. Recall that in Proof D.1.2, r and s remains same in both q_0 and q_1, as in binary representation all mkws and qs have the same length n. However, the number of ws in q changes, and consequently, number of mkws change. Hence, the range of indices k_0 and k_1 are different for q_0 and q_1. This does not affect r and s which are positions of keywords (not related to number of keywords) in the binary representation of fixed length. Hence, the same argument from the proof of Lemma D.2 holds. □