CONJURA · OPEN PROBLEM

Level-Optimal Search Trees Beyond Disjunctive Queries

Balanced OR trees against the unbalanced optimum

Status. Statement: AI-written from Mohammad Etemad, Mohammad Mahmoody and David Evans, *Optimizing Trees for Static Searchable Encryption*, IACR Cryptology ePrint Archive 2018, not yet checked by a human. Proved for every distribution supported on disjunctive queries, where the $\log n$ factor is also shown to be necessary; open for every larger class, and in particular for distributions over conjunctive queries, where the paper's proof technique demonstrably fails.

Category. *Searchable Encryption & Encrypted Data Structures.*

Conjecture (informal). *Tree-based encrypted search stores each file at a leaf of a binary tree, tagged with the set of keywords it contains, and tags every internal node with the union of the keyword sets below it. A query is answered by walking down from the root, descending only below nodes whose tag is consistent with the query, so the cost of a search is the number of nodes touched — and that number depends entirely on which files were placed next to each other when the tree was built. The paper gives an algorithm that builds the tree bottom-up, one level at a time, pairing up the current nodes by a minimum-weight perfect matching whose edge weights are the probability that a random query is satisfied by the merged node. The resulting tree is balanced, which is what you want for encrypted search, because an unbalanced tree leaks its own shape and costs more rounds of interaction. The authors prove that when every query is a disjunction of keywords, this balanced tree is never worse than a logarithmic factor times the cheapest tree of any shape, and they show a family of file sets where that logarithmic loss really is unavoidable. They conjecture the same guarantee survives for a wider class of query distributions. The reason this is not routine is that their proof never looks at the algorithm at all: for disjunctions, every balanced tree is automatically within a logarithmic factor, because a node can only be consistent with a disjunctive query if some matching file sits beneath it. That fails as soon as conjunctions are allowed — a node can be consistent with an AND of two keywords while no single file below it contains both — and the paper exhibits a file arrangement where an AND query sweeps the entire tree and returns nothing. So for conjunctions no shape-only argument can work, and any proof has to use something about the matchings the algorithm actually picks.*

1 The setting

Tree-based searchable symmetric encryption stores the encrypted index as a binary search tree [Goh03, KP13, BlindSeer]: each file sits at a leaf together with a vector saying which dictionary keywords it contains, and each internal node carries the bitwise OR of its children's vectors, so that a node's vector describes the union of the keywords of the files in its subtree. A monotone Boolean query q is answered by starting at the root and descending below exactly

those visited nodes whose label satisfies q; the satisfying leaves are the answer. Because every node is processed by the same constant-time (or same secure-computation) procedure, the search cost is proportional to the number of visited nodes, and this is the quantity that all of these schemes pay per query. The leakage of such a scheme is the usual access and search pattern [CGKO06] plus the set of visited nodes, which the paper calls the tree pattern.

The observation driving the paper is that previous tree-based schemes assign files to leaves arbitrarily, while their security proofs do not depend on that assignment — so the assignment is free to be optimized. Fix a distribution Q over monotone queries and write $V_T(q)$ for the number of nodes visited when searching q in a tree T. The paper shows (its Lemma B.3) that for any OR tree T over L,

$$\begin{aligned} \mathop{\mathbb{E}}_{q \leftarrow Q}\left[V_T(q) \right] &= 1 + 2 \sum_{u \in I(T)} \mathsf{P}_Q(u) \\ &= 1 + 2\,\mathrm{SLC}_Q(T), \end{aligned}$$

which turns tree design into the combinatorial problem of choosing the file grouping that minimizes the sum of the local costs $\mathsf{P}_Q(u)$ over the $n-1$ internal nodes. The paper gives three heuristics for this: a Huffman-style greedy one (best-first) that produces unbalanced trees, the level-optimal algorithm of Definition 2, which pairs each level up by a minimum-weight perfect matching computed with Edmonds' blossom algorithm [Edm65], and a cheaper hybrid. Balance is not cosmetic here: an unbalanced tree leaks its own shape at build time, has worst-case search cost linear in n, and costs more rounds when the node processing is done by secure computation as in Blind Seer [BlindSeer]. So the question is how much one loses by insisting on a balanced tree and on a level-by-level greedy construction of it.

What is known. The paper proves that when Q is supported on disjunctive queries, $\mathrm{SLC}_Q(\mathsf{LO}(L,Q)) = O(\log n \cdot \mathrm{Opt}_Q(L))$ (its Theorem B.1), and that the logarithmic factor cannot be improved to $o(\log n)$: for n files with pairwise disjoint keyword sets in which the i-th file has 2^{i-1} keywords, so that $m = 2^n - 1$, and for Q uniform over single keywords, the best-first solution costs less than $2^{n+2} \approx 4m$ while the level-optimal solution costs $2(2^n - 1)\log n \approx 2m \log n$, so the ratio of the level-optimal cost to the best-first cost is $\Omega(\log n)$. Separately, for Q uniform over single keywords, n even, and under

the density hypothesis $\mathrm{SLC}_Q(\mathsf{LO}(L,Q)) = \alpha(n-1)$ with $\alpha > 1/2$, the paper proves the sharper bound

$$\mathrm{SLC}_Q(\mathsf{LO}(L,Q)) \;\le\; \frac{\alpha}{2\alpha-1}\,\mathrm{OptBal}_Q(L),$$

where OptBal is the optimum over balanced trees only (its Theorem B.2); its proof works layer by layer, showing that the parents of the leaves chosen by the matching already carry at least half the zero-weight of any balanced tree. Nothing is proved for conjunctive or general monotone Q.

Experimentally (Enron corpus, queries that are single keywords and conjunctions and disjunctions of two and three keywords), the level-optimal and best-first algorithms are compared for $n \le 2^{10}$, where their costs stay close; for $n > 2^{10}$, up to $n = 2^{14}$ in the paper's Figure 5, the cheaper hybrid algorithm is used in place of the level-optimal one. On $n = 8$ leaves, exhaustive search over all *balanced* trees finds a tree better than the level-optimal one in 4 of 25 experiment sets, with worst-case difference 0.07 in cost, while exhaustive search over all 135,135 trees finds a tree better than the *best-first* one in 13 of 25, with difference 0.09.

Why the disjunctive proof stalls. The paper's Theorem B.1 argument never inspects the algorithm: for a disjunctive $q = \bigvee_{j\in S} x_j$, a node's label satisfies q only if some satisfying leaf lies beneath it, so the satisfying nodes of *any* tree of depth d lie on the $\le d$-long root paths of the $|S|$ satisfying leaves, giving $\mathrm{SLC}_q(T_d) \le |S|\cdot d$ against $\mathrm{SLC}_q(T) \ge |S|$ for every T. That is a bound on every balanced tree, not on the algorithm's output. It collapses for conjunctions, and the paper says why: a node can satisfy $w_1 \wedge w_2$ while the two keywords are contributed by different children, so no file below it satisfies the query. See Remark 3.

2 Notation and parameters

$[n] = \{1,\dots,n\}$, and log is to base 2.

The dictionary is a set of m keywords $w_1,\dots,w_m$. A vector $\bar v \in \{0,1\}^m$ records a set of keywords, with $\bar v[j] = 1$ meaning w_j is present; $\mathrm{OR}(\bar x,\bar y) \in \{0,1\}^m$ is the bitwise OR, i.e. $\mathrm{OR}(\bar x,\bar y)[j] = \bar x[j] \vee \bar y[j]$ for all $j \in [m]$.

$\mathcal{M}_m$ is the set of monotone Boolean formulas over variables $x_1,\dots,x_m$: formulas built from the variables using $\wedge$ and $\vee$ only,

with no negation. For $q \in \mathcal{M}_m$ and $\bar{v} \in \{0,1\}^m$ we write $q(\bar{v}) = 1$ if q evaluates to true under the assignment $x_j := \bar{v}[j]$ for all $j \in [m]$, and $q(\bar{v}) = 0$ otherwise. A *disjunctive* query is one of the form $\bigvee_{j \in S} x_j$ and a *conjunctive* query one of the form $\bigwedge_{j \in S} x_j$, for $S \subseteq [m]$.

$L = \{(f_1, \bar{f_1}), \ldots, (f_n, \bar{f_n})\}$ is a set of n file identifiers f_i with vector labels $\bar{f_i} \in \{0,1\}^m$, where $\bar{f_i}[j] = 1$ iff file f_i contains keyword w_j. For a tree T, $L(T)$ is its set of leaves, $I(T)$ its set of internal nodes, $\pi(u)$ the parent of a non-root node u, and $\Gamma(u)$ the set of children of an internal node u. Each node u of T carries a label $\bar{u} \in \{0,1\}^m$.

Q denotes a distribution over $\mathcal{M}_m$, and $q \leftarrow Q$ a sample from it. For a node u with label $\bar{u}$, its *local cost* is

$$\mathsf{P}_Q(u) \;=\; \Pr_{q \leftarrow Q}\big[\, q(\bar{u}) = 1 \,\big] \;\in\; [0,1].$$

For an OR tree T (Definition 1) the *sum of local costs* and the *optimum* over all OR trees on the same leaves are

$$\mathrm{SLC}_Q(T) \;=\; \sum_{u \in I(T)} \mathsf{P}_Q(u),$$

$$\mathrm{Opt}_Q(L) \;=\; \min_{T\,:\,L(T)=L} \mathrm{SLC}_Q(T),$$

the minimum in $\mathrm{Opt}_Q(L)$ ranging over all OR trees over L, balanced or not; $\mathrm{OptBal}_Q(L)$ denotes the same minimum restricted to balanced trees. Finally $\mathsf{LO}(L,Q)$ is the set of trees the level-optimal algorithm of Definition 2 can output on L and Q.

The parameters are:

- m, the dictionary size, i.e. the number of keywords; labels are vectors in $\{0,1\}^m$. Unbounded and independent of n.
- $n = 2^k$, the number of files, hence the number of leaves; a power of two, as the level-optimal algorithm requires, with $k \geq 1$.
- L, the leaf set: n file identifiers with their keyword vectors, arbitrary.
- Q, the query distribution, a distribution over monotone Boolean formulas on m variables.

- c, the approximation constant, absolute: independent of m, n, L and Q.

Definition 1 (OR tree over L). Let $L = \{(f_1, \bar{f}_1), \ldots, (f_n, \bar{f}_n)\}$ with $\bar{f}_i \in \{0,1\}^m$. An *OR tree over L* is a full binary tree T (every node has either 0 or 2 children) with $L(T) = L$, in which every leaf f_i carries its given label $\bar{f}_i$ and every internal node $u \in I(T)$ with $\Gamma(u) = \{x, y\}$ carries the label $\bar{u} = \mathrm{OR}(\bar{x}, \bar{y})$. The labels are therefore determined by L together with the shape of T and the assignment of files to leaves; $|I(T)| = n - 1$ always. We call T *balanced* if its height is at most $\lceil \log n \rceil$.

Definition 2 (the level-optimal algorithm LO). On input a leaf set L with $|L| = n = 2^k$ and access to the local cost function $\mathsf{P}_Q(\cdot)$, the algorithm sets $W := L$ and, while $|W| > 1$, does the following. Form the complete graph on vertex set W and give each edge $\{x, y\}$, $x \neq y$, the weight $C(x, y) := \mathsf{P}_Q(z)$ where z is a new node with label $\bar{z} = \mathrm{OR}(\bar{x}, \bar{y})$. Compute a perfect matching M on this graph, i.e. a set of $|W|/2$ disjoint edges, of minimum total weight $\sum_{\{x,y\} \in M} C(x, y)$. For every $\{x, y\} \in M$, create the node z with label $\bar{z} = \mathrm{OR}(\bar{x}, \bar{y})$, set $\Gamma(z) = \{x, y\}$ and $\pi(x) = \pi(y) = z$, remove x and y from W and add z to W. When $|W| = 1$ return the tree built, whose root is the single remaining element of W; it is a balanced OR tree over L of height exactly k. We write $\mathsf{LO}(L, Q)$ for the set of all trees obtainable this way, over all choices of minimum-weight perfect matching at each level.

3 The conjecture

Conjecture 1 (level-optimal beyond disjunctions). *There are a class C of query distributions, strictly containing every distribution supported on disjunctive queries, and an absolute constant $c > 0$, such that the following holds for every dictionary size $m \geq 1$, every $k \geq 1$ with $n = 2^k$, every leaf set $L = \{(f_1, \bar{f}_1), \ldots, (f_n, \bar{f}_n)\}$ with $\bar{f}_i \in \{0,1\}^m$, every $Q \in C$ over $\mathcal{M}_m$, and every tree $T \in \mathsf{LO}(L, Q)$:*

$$\mathrm{SLC}_Q(T) \;\leq\; c \cdot \log n \cdot \mathrm{Opt}_Q(L).$$

Equivalently, $\mathrm{SLC}_Q(T) = O(\log n \cdot \mathrm{Opt}_Q(L))$ *with the hidden constant independent of* m, n, L *and* Q.

The paper does not name C: it writes only that it conjectures "a more general theorem (than our Theorem B.1) holds for a broader class of query distributions". Identifying such a class is part of the problem. For every Q supported on disjunctions $\bigvee_{j\in S} x_j$, $S \subseteq [m]$, the displayed bound is a theorem of the paper; the content of the conjecture is that C can be taken strictly larger.

Conjecture 2 (maximal formalization). *Conjecture 1 holds with* C *the class of* all *distributions over* $\mathcal{M}_m$.

Conjecture 2 is not the paper's statement. It is the largest class available in the paper's own framework — the class over which the algorithms are defined and where the experiments live — and it is recorded here as the maximal formalization of the paper's sentence, not as a claim about what the paper asserts.

Remark 1 (the conjunctive restriction). The intermediate reading worth naming is Conjecture 1 with C the distributions supported on conjunctive queries $\bigwedge_{j\in S} x_j$. That is the case the paper repeatedly singles out as the hard and interesting one, it is where the experiments of Section 1 live alongside the disjunctive ones, and settling it would discharge the paper's intent. A reviewer may reasonably prefer it to Conjecture 2.

Remark 2 (a deliberate strengthening: all matchings). The paper's Definition B.2 writes $T_{\mathsf{LO}}(L)$ as a single tree. The minimum-weight perfect matching of Definition 2 need not be unique, so the algorithm's output is not either, and the conjectures above quantify over *every* $T \in \mathsf{LO}(L, Q)$. This is a deliberate strengthening of the paper's formulation. A proof that covers only some canonical tie-breaking rule would be a partial answer, and worth reporting as such.

Remark 3 (the conjecture is about the algorithm, not about balance). For disjunctive Q the paper's bound happens to hold for every balanced tree, by the argument recalled in Section 1. For conjunctive Q that is false. Take n files, half of them carrying $\{Female, Dept1\}$ and half carrying $\{Male, Dept2\}$, placed at alternate

leaves of a balanced tree. The query *Female* $\wedge$ *Dept2* is satisfied at every internal node — each has both keyword sets beneath it — so it sweeps the whole tree and returns nothing, whereas grouping the two halves under the two children of the root, also a balanced tree, makes only the root satisfy the query. So for conjunctive Q the gap between an arbitrary balanced tree and $\mathrm{Opt}_Q(L)$ can be $\Omega(n)$. Hence the quantification over $T \in \mathsf{LO}(L,Q)$ rather than over balanced trees is forced if the statement is to be non-trivial, and any proof must use a property of the matchings that Definition 2 actually computes rather than only the height of the tree.

4 Bibliography

[BlindSeer] V. Pappas, F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S. G. Choi, W. George, A. Keromytis, S. Bellovin. *Blind Seer: A Scalable Private DBMS.* 35th IEEE Symposium on Security and Privacy, 2014.

[CGKO06] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky. *Searchable symmetric encryption: improved definitions and efficient constructions.* ACM CCS'06, 2006.

[Edm65] J. Edmonds. *Paths, Trees, and Flowers.* Canadian Journal of Mathematics, 17(3):449–467, 1965.

[Goh03] E.-J. Goh. *Secure indexes.* Technical report, Cryptology ePrint Archive, Report 2003/216, 2003.

[KP13] S. Kamara, C. Papamanthou. *Parallel and dynamic searchable symmetric encryption.* Financial Cryptography and Data Security, FC, 2013.