The Planted Orthogonal Vectors Problem

David Kühnemann* Adam Polak† Alon Rosen‡

Abstract

In the k-Orthogonal Vectors (k-OV) problem we are given k sets, each containing n binary vectors of dimension $d = n^{o(1)}$, and our goal is to pick one vector from each set so that at each coordinate at least one vector has a zero. It is a central problem in fine-grained complexity, conjectured to require $n^{k-o(1)}$ time in the worst case.

We propose a way to *plant* a solution among vectors with i.i.d. p-biased entries, for appropriately chosen p, so that the planted solution is the unique one. Our conjecture is that the resulting k-OV instances still require time $n^{k-o(1)}$ to solve, *on average*.

Our planted distribution has the property that any subset of strictly less than k vectors has the *same* marginal distribution as in the model distribution, consisting of i.i.d. p-biased random vectors. We use this property to give average-case search-to-decision reductions for k-OV.

1 Introduction

The security of cryptographic systems crucially relies on heuristic assumptions about average-case hardness of certain computational problems. Sustained cryptanalysis alongside technological advances such as large-scale quantum computers, put these hardness assumptions under constant risk of being invalidated. It is therefore desirable to try to design cryptographic schemes based on new computational problems, preferably ones whose hardness is well-studied.

The field of computational complexity developed over the last fifty years a good understanding of hardness of certain problems – e.g., SAT is widely believed to require at least superpolynomial, maybe even exponential time [19] – however these are worst-case problems, and hence unsuitable for direct use as a basis for cryptography.

Fine-grained complexity [18] is a younger branch of computational complexity that studies "hardness of easy problems", i.e., problems known to be solvable in polynomial time but supposedly not faster than some specified polynomial, say not faster than in cubic time. It gives rise to *fine-grained cryptography* [5, 15, 17]; the idea that it might be possible to build cryptography, notably public-key encryption, based on conjectured average-case hardness of polynomial-time problems studied in fine-grained complexity. These problems are easier than NP-hard ones, but for polynomials of sufficiently high degree may still be hard enough to give honest parties an adequate advantage over malicious attackers.

*University of Amsterdam. david.kuhnemann@student.uva.nl. Part of this work done while visiting Bocconi.

†Bocconi University. adam.polak@unibocconi.it.

‡Bocconi University. alon.rosen@unibocconi.it. Work supported by European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme (Grant agreement No. 101019547) and Cariplo CRYPTONOMEX grant.

1.1 The k-Orthogonal Vectors Problem

The Orthogonal Vectors (OV) problem [20], together with its generalization k-OV, is one of the three main hard problems studied in fine-grained complexity, alongside 3SUM and APSP [18]. Arguably, among the three, OV is the one whose (worst-case) hardness we understand the most – in particular because it is implied by the Strong Exponential Time Hypothesis (SETH) [11], which is about the very well studied SAT problem.

We say that vectors $u_1, ..., u_k \in \{0,1\}^d$ are *orthogonal* if, for all $j \in [d]$, $\prod_{\ell=1}^{k} u_\ell[j] = 0$, meaning that for every coordinate there is at least one zero entry among the k vectors. For $k \geq 2$, let $U_1, ..., U_k \in \{0,1\}^{n \times d}$ be collections of n d-dimensional binary vectors, where $U_{\ell,i}$ denotes the i-th vector of U_ℓ. The k-Orthogonal Vectors problem (k-OV) asks whether there exist $(s_1, \ldots, s_k) \in [n]^k$ such that $U_{1,s_1}, U_{2,s_2}, \ldots, U_{k,s_k}$ are orthogonal.

Worst-case complexity. The naive algorithm solves k-OV in time $O(n^k d)$. For any fixed constant c, the algorithms by Abboud et al. [1] and Chan and Williams [6] solve OV in dimension $d = c \log n$ in time $O(n^{2-\varepsilon_c})$ for $\varepsilon_c > 0$. However, Gao et al. [10] conjecture that no such strongly subquadratic algorithm exists for superlogarithmic dimension $d = \omega(\log n)$. This conjecture (known as Low-dimension Orthogonal Vector Conjecture) is also implied by SETH [20]. Both the upper bound for $d = O(\log n)$ and the SETH-implied hardness for $d = \omega(\log n)$ generalize to k-OV, for any constant $k \geq 2$, where the running time barrier is n^k [18].

Average-case complexity. For cryptographic purposes we care about *average-case* hardness – because we want to be able to efficiently sample instances that are hard to solve (in contrast to only having an existential knowledge that there are some hard instances). Moreover, the sampler shall correctly tell (with high probability) whether its output is a YES- or NO-instance.

One way to achieve this is to embed a solution in an instance sampled from a distribution that generates NO-instances with high probability. This method of *planting* a solution has been applied to a number of problems, e.g., k-Clique [13] and (in the fine-grained setting) k-SUM [9, 2] (a generalization of 3SUM) and Zero-k-Clique [15] (a generalization of Zero-Triangle, which is a problem harder than both 3SUM and APSP), but not for (k-)OV. The following question remains wide-open [5, 8, 7]:

How to plant orthogonal vectors (so that they are hard to find)?

1.2 Our results

We propose a way of planting a solution in k-OV instances where each vector entry is i.i.d. according to a p-biased[1] coin flip, for an appropriately chosen value of p so that the planted solution is the only one in the instance, with good probability. We conjecture that solving these instances require $n^{k-o(1)}$ time on average.

Superlogarithmic dimension. The k-OV problem might have appeared as a poor candidate for a fine-grained average-case hard problem, as Kane and Williams [14] showed that for any fixed $p \in (0,1)$, k-OV instances of i.i.d. p-biased entries can be solved in $O(n^{k-\varepsilon_p})$ time for some $\varepsilon_p > 0$ by AC0 circuits. However, such instances are only non-trivial for $d = \Theta(\log n)$, a parameter setting

[1]We say that a random bit is p-biased if it equals 1 with probability p and equals 0 with probability $1-p$.

which can be anyway solved in time $O(n^{k-\varepsilon})$, even in the worst case, using the algorithm of Chan and Williams [6]. To obtain a candidate hard distribution based on i.i.d. entries, we therefore choose to sample the entries as 1 with subconstant probability $p(n) = o(1)$, which leads to nontrivial instances in the superlogarithmic dimension regime $d = \omega(\log n)$. In Appendix B we present another simple argument why a logarithmic dimension is not sufficient, further justifying our choice.

$(k-1)$-wise independence. Our planting procedure has the following notable property: any subset of $k-1$ (or less) out of the k vectors that form the planted solution has the marginal distribution identical to that of $k-1$ independently sampled vectors with i.i.d. p-biased random entries. In particular, each individual vector of the solution has the same marginal distribution as any other vector in the instance. This would not be true if we planted k random vectors conditioned on orthogonality (i.e., a type of solution that may appear spontaneously with small probability), because such vectors tend to be sparser than the expectation. This sparsity is what makes the Kane–Williams algorithm [14] work, and lack thereof makes our instances immune to that algorithm.

We note that the $(k-1)$-wise independence property holds “for free” in natural distributions for k-SUM [2, 9] and Zero-k-Clique [15] because of the natural symmetry of cyclic groups $\mathbb{Z}_m$. However, it is a priori unclear how to get it for k-OV.

Search-to-decision reductions. To demonstrate the usefulness of the $(k-1)$-wise independence property, we give a fine-grained average-case search-to-decision reduction for our conjectured hard k-OV distribution. Actually, we give two such reductions, for two different parameter regimes. The simpler one, in Section 6, introduced an $O(\log n)$ overhead in the failure probability, so it is relevant only if the decision algorithm succeeds with probability higher than $1 - \frac{1}{\log n}$. The other reduction, in Section 7, works with small constant failure probabilities on both ends.

Planting multiple solutions. We also argue that $(k-1)$-wise independence allow planting more than one solution in a single instance, which we believe might be useful for building cryptographic primitives.

Let us remark that all our results are nontrivial already for $k = 2$, i.e., for the Orthogonal Vectors problem. However, from the point of view of cryptographic applications, larger values of k are more interesting (as they potentially offer a bigger advantage for the honest parties), so we present all our results in full generality.

1.3 Technical overview

Planting. How do we generate k orthogonal vectors such that any $k-1$ of them look innocent? First of all, we can focus on generating a single coordinate, and then repeat the process independently for each of the d coordinates. Consider the joint distribution of k i.i.d. p-biased random bits. We need to modify it to set the probability of k ones to 0. If we just do it, and scale up the remaining probabilities accordingly, the probability of $k-1$ ones turns out wrong. After we fix that, the probability of $k-2$ ones is off, and so on, in a manner similar to the exclusion-inclusion principle. By doing this mental exercise we end up with a formula for the joint distribution of k bits in a single coordinate of the k vectors to be planted. How do we actually sample from this distribution? Since it has the $(k-1)$-wise independence property, the following approach must work:

First sample $k-1$ i.i.d. p-biased bits, and then sample the k-th bit with probability depending on the number of ones among the first $k-1$ bits. In Section 3 we show how to set this last probability exactly.

Search-to-decision reductions. Both our reductions are based on the same basic idea: In order to find the planted solution, we replace some of the vectors in the input instance with newly sampled vectors with i.i.d. p-biased entries and run the decision algorithm on such a modified instance. If at least one of the planted vectors got resampled, the resulting instance has the same distribution as if no planting occurred (thanks to the $(k-1)$-wise independence), and so the decision algorithm returns NO with good probability. Otherwise the planted solution is still there and the decision algorithm likely says YES.

Our first reduction (see Section 6) applies this idea to perform a binary search. It introduces a factor of $k \log n$ overhead in the running time and also in the failure probability, because we need to take a union bound over all invocations of the decision algorithm returning correct answers.

Our second reduction (see Section 7) is an adaptation of a search-to-decision reduction for k-SUM due to Agrawal et al. [2]. In short, the reduction repeatedly resamples a random subset of vectors, runs the decision algorithm, and keeps track for each of the original vectors, how many times the decision algorithm returned YES when this vector was not resampled. Statistically, this count should be larger for vectors in the planted solution. A careful probabilistic analysis shows that this is indeed the case.

1.4 Open problems

Fine-grained asymmetric cryptography. A key goal of fine-grained cryptography is to devise an advanced asymmetric cryptography scheme – such as public key encryption – whose security is based on hardness of a well understood problem from fine-grained complexity. So far the closest to this goal seems to be the key exchange protocol due to LaVigne, Lincoln, and Vassilevska Williams [15], which is based on hardness of the Zero-k-Clique problem. Despite being based on a parameterized problem (that allows for arbitrary polynomial $n^{k-o(1)}$-hardness by simply choosing a large enough k), the protocol offers only quadratic security, i.e., breaking the encryption takes only quadratically more than it takes to encrypt and decrypt a message. This limitation seems inherent to the protocol because it is based on a similar idea as Merkle puzzles [16].

It is an open problem if fine-grained cryptography with superquadratic security is possible. We believe that k-OV could be a good hard problem for that purpose, because of a different structure, which addition-based problems, like k-SUM and Zero-k-Clique, are lacking.

In recent work, Alman, Huang, and Yeo [4] show that if one-way functions do not exist then average-case hardness fine-grained assumptions on planted k-SUM and Zero-k-Clique are false for sufficiently large constant k. Being a planted problem, the same is expected to apply to k-OV [4]. A construction of public-key encryption from planted k-OV would be interesting even in a world where one-way functions do exist, as they are not known to imply public-key encryption [12].

Faster algorithms for average-case OV. Algorithms for random OV instances seem to be underexplored. Up until recently [3] it was not known if the average-case OV admits even a subpolynomial improvement compared to the worst case. With this paper we hope to inspire more research in this direction. We would even be happy to see our conjecture refuted.

2 The model distribution

Fix $k \geq 2$, and let $d = \alpha(n) \log n$ for $\alpha(n) = \omega(1)$. We define the family of *model distributions* $k\text{-OV}_0^\alpha(n)$ that generate k matrices $U_1, ..., U_k \in \{0,1\}^{n \times d}$ where all entries are i.i.d. p-biased bits with probability

$$p = \left(1 - 2^{-\frac{2k}{\alpha(n)}}\right)^{\frac{1}{k}} \approx \left(\frac{2k \ln(2)}{\alpha(n)}\right)^{\frac{1}{k}}.$$

As it becomes apparent later, for the planting algorithm to work it is crucial that $p \leq 1/2$, but thanks to $\alpha(n) = \omega(1)$ this holds for large enough n.

We show that the model distribution indeed generates NO-solutions with good probability.

Lemma 1. *A k-OV instance sampled from the model distribution $k\text{-OV}_0^\alpha(n)$ is a* NO*-instance with probability at least $1 - \frac{1}{n^k}$.*

Proof. For a k-OV instance $\mathbf{U} = (U_1, \ldots, U_k) \sim k\text{-OV}_0^\alpha(n)$, a fixed combination of vectors $u_1, \ldots, u_k$ (where $u_\ell \in U_\ell$) is orthogonal iff, for each coordinate $j \in [d]$, not all of the k vectors have one in that coordinate. Since k i.i.d. p-biased bits are all ones with probability p^k, the probability that $u_1, \ldots, u_k$ are orthogonal (determined by the all-ones event not occurring in any of the d coordinates) is:

$$\Pr[u_1, ..., u_k \text{ are orthogonal}] = \left(1 - p^k\right)^d = \left(2^{-\frac{2k}{\alpha(n)}}\right)^{\alpha(n)\log(n)} = n^{-2k}.$$

By linearity of expectation, the expected value for the number of solutions among all n^k possible combinations of k vectors, denoted by $c(\mathbf{U})$, is

$$\mathbb{E}[c(\mathbf{U})] = \sum_{\substack{u_i \in U_i \\ (1 \leq i \leq k)}} \Pr[u_1, \ldots, u_k \text{ are orthogonal}] = n^k \cdot n^{-2k} = \frac{1}{n^k}.$$

This gives us a bound on the probability of *any* solution occurring:

$$\Pr[c(\mathbf{U}) > 0] = \sum_{i=1}^{n^k} \Pr[c(\mathbf{U}) = i] \leq \sum_{i=1}^{n^k} i \cdot \Pr[c(\mathbf{U}) = i] = \mathbb{E}[c(\mathbf{U})] = \frac{1}{n^k}.$$

Hence, an instance sampled from $k\text{-OV}_0^\alpha(n)$ is a NO-instance with probability $1 - \frac{1}{n^k}$. □

We remark that one can make the probability of sampling a NO-instance arbitrarily high; in order to get the probability $1 - \frac{1}{n^c}$ it suffices to replace $2k$ with $k+c$ in the formula for the probability parameter p. However, having in mind the cryptographic motivation, $\frac{1}{n^k}$ seems a reasonable default choice for the failure probability of the sampler, because with the same probability the attacker can just guess the solution.

3 The planted distribution

To plant a solution at locations $s_1, \ldots, s_k \in [n]$ in an instance $\mathbf{U}$ sampled from $k\text{-OV}_0^\alpha(n)$, we apply the following randomized algorithm.

$\mathsf{Plant}(\mathbf{U}, s_1, \ldots, s_k)$:

1. For each coordinate $1 \le j \le d$:

   (a) Let m be the number of ones among $U_{1,s_1}[j], \dots, U_{k,s_k}[j]$.

   (b) If $m - k$ is even, flip $U_{k,s_k}[j]$ with probability $\left(\frac{p}{1-p}\right)^{k-m}$. (Here we need $p \le 1/2$.)

2. Return $\mathbf{U}$.

We justify this way of planting in Section 4. For now, observe that if all vectors $U_{1,s_1}, \dots, U_{k,s_k}$ feature a one at coordinate j, we have $m = k$ and Plant flips the final bit $U_{k,s_k}[j]$ to a zero with probability

$$\left(\frac{p}{1-p}\right)^{k-m} = \left(\frac{p}{1-p}\right)^{0} = 1.$$

Thus, $\mathsf{Plant}(\mathbf{U}, s_1, \dots, s_k)$ outputs a YES-instance of k-OV with a solution at $s_1, \dots, s_k$.

We sample YES-instances of k-OV by planting a solution in an instance $\mathbf{U} \sim k\text{-OV}_0^\alpha(n)$ at locations $s_1, \dots, s_k$ chosen uniformly at random.

Distribution $k\text{-OV}_1^\alpha(n)$:

1. Sample $\mathbf{U}$ from $k\text{-OV}_0^\alpha(n)$.

2. Sample $(s_1, \dots, s_k)$ uniformly at random from $[n]^k$.

3. Return $\mathsf{Plant}(\mathbf{U}, s_1, \dots, s_k)$.

The above observation about Plant immediately yields the following.

Lemma 2. *A k-OV instance sampled from the planted distribution $k\text{-OV}_1^\alpha(n)$ is a YES-instance with probability* 1.

4 $(k-1)$-wise independence of planted vectors

Our method of planting orthogonal vectors arises from the idea that for any planted problem, any proper "piece" of the planted solution should be indistinguishable from any comparable piece of the instance as a whole, conditioned on the latter still being consistent with being a part of a solution itself.

For example, in the case of planting a k-clique in a graph G this requirement is trivial. Indeed, the projection of the clique onto a smaller subset of $k' < k$ vertices yields a k'-clique, which are exactly those subgraphs of G of size k' which could feasibly belong to a solution.

In contrast to the previous example, in the case of k-SUM, any set of $k-1$ elements $x_1, \dots, x_{k-1}$ in an instance could feasibly be part of a solution, as one can always construct a $k-$th number x_k such that $\sum_{i=1}^k x_i = 0$. Thus, by the principle we described, to plant a solution in an instance with i.i.d. uniformly random elements, the marginal distribution of the distribution of planted solutions $(x_1, \dots, x_k)$ given by any projection to $k-1$ elements should itself be uniformly random. This holds true in the case of the planted k-SUM [2], where the planted solution is distributed uniformly over the set of all k-tuples that form valid k-SUM solutions. The case of planted Zero-k-Clique [15] is analogous. For both of these problems, k elements from the model distribution conditioned on them feasibly forming a solution yields a distribution that is suitable for planting according to the described principle.

This is different from the k-OV problem with a model distribution of i.i.d. vector entries. Here, as with k-SUM and Zero-k-Clique, any set of $k-1$ elements (in this case vectors) could form a solution to k-OV. All that is needed is for the last vector to feature a 0 in all those coordinates where the other $k-1$ all were 1. However, sparse vectors are far more likely to be part of a solutions than dense ones. Therefore, conditioning k i.i.d. p-biased vectors on being orthogonal yields a distribution which does not follow our principle; Projecting onto any subset of $k' < k$ vectors results in vectors that are on average sparser than (and thus different from) k' i.i.d. p-biased vectors. As we will show now, our method of planting *does* satisfy this principle: Any subset of $k-1$ planted vectors are independent and identically distributed p-biased vectors.

Let $M \sim k\text{-OV}_0^\alpha(n)$ and $\mathbf{U} = \mathsf{Plant}(M, s_1, \ldots, s_k)$. Recall that both sampling from the model distribution $k\text{-OV}_0^\alpha(n)$ and the planting by Plant are independent and identical for each coordinate $j \in [d]$. Hence, all k-bit sequences $x = (U_{1,s_1}[j], U_{2,s_2}[j], \ldots, U_{k,s_k}[j]) \in \{0,1\}^k$, for all $j \in [d]$, are independent and identically distributed, according to a distribution whose probability density function we denote by $\mathcal{P}_k : \{0,1\}^k \to \mathbb{R}$.

Lemma 3. *Let $x \in \{0,1\}^k$ contain m ones, i.e., $||x||_1 = m$. Then*

$$\mathcal{P}_k(x) = p^m(1-p)^{k-m} - (-1)^{k-m}p^k.$$

Proof. Fix a coordinate $j \in [d]$. Let $X = (M_{1,s_1}[j], M_{2,s_2}[j], \ldots, M_{k,s_k}[j])$ be the random variable denoting the entries of the j-th coordinate among the vectors at locations $s_1, \ldots, s_k$ before planting. We proceed by case distinction.

Case 1. If $m-k$ is even, the probability of x occurring in the given coordinate $j \in [d]$ of the planted solution is given by

$$\begin{aligned}
\mathcal{P}_k(x) &= \Pr[X = x \text{ and } \mathsf{Plant} \text{ does not flip the final bit}] \\
&= p^m(1-p)^{k-m} \cdot \left(1 - \left(\frac{p}{1-p}\right)^{k-m}\right) \\
&= p^m(1-p)^{k-m} - 1 \cdot p^k \\
&= p^m(1-p)^{k-m} - (-1)^{k-m}p^k.
\end{aligned}$$

Case 2a. If $m-k$ is odd and $x = y1$ for some $y \in \{0,1\}^{k-1}$, then $x = y1$ may occur either directly in the model instance, or by $y0$ (for which $m-k$ is even) occurring in the model instance and Plant flipping the final bit:

$$\begin{aligned}
\mathcal{P}_k(x) &= \Pr[X = y1] + \Pr[X = y0 \text{ and } \mathsf{Plant} \text{ flips the final bit}] \\
&= p^m(1-p)^{k-m} + p^{m-1}(1-p)^{k-(m-1)} \cdot \left(\frac{p}{1-p}\right)^{k-(m-1)} \\
&= p^m(1-p)^{k-m} + p^k \\
&= p^m(1-p)^{k-m} - (-1)^{k-m}p^k.
\end{aligned}$$

Case 2b. Similarly, if $m-k$ is odd and $x = y0$ for some $y \in \{0,1\}^{k-1}$, then $x = y0$ can occur either directly in the model instance or by Plant flipping the final bit of the sequence $y1$:

$$\begin{aligned}\mathcal{P}_k(x) =\ & \Pr\left[X = y0\right] + \Pr\left[X = y1 \text{ and Plant flips the final bit}\right] \\ &= p^m(1-p)^{k-m} + p^{m+1}(1-p)^{k-(m+1)} \cdot \left(\frac{p}{1-p}\right)^{k-(m+1)} \\ &= p^m(1-p)^{k-m} + p^k \\ &= p^m(1-p)^{k-m} - (-1)^{m-k}p^k.\end{aligned}$$

□

Remark 1. *It follows immediately from Lemma 3 that the method of planting is invariant under permutations of the k collections $U_1, \ldots, U_k$.*

Having $\mathcal{P}_k$ as the distribution of planted vectors, rather than, e.g., the k-vector joint model distribution conditioned on orthogonality, ensures $(k-1)$-*wise independence* among the planted vectors. I.e., the projection of k planted vectors onto any subset of size $k' < k$ is identically distributed to k' vectors from the model distribution.

Lemma 4 ($(k-1)$-wise independence)**.** *Marginalizing any one of the k bits of $\mathcal{P}_k$ yields $k-1$ independent p-biased bits.*

Proof. By Remark 1 we may assume w.l.o.g. that the last bit is the one marginalized out. The lemma then follows from the definition of Plant, as the first $k-1$ entries of any coordinate in the planted vectors are unchanged from the model instance, and are therefore independent p-biased bits. □

This property is useful in bounding the probability of a planted instance containing a solution besides the planted one.

Lemma 5. *A k-OV instance sampled from the planted distribution $k\text{-}OV_1^{\alpha}(n)$ has more than one solution with probability less than $\frac{1}{n^k}$.*

Proof. While the k planted vectors are guaranteed to form a solution, by $(k-1)$-wise independence, *all* combinations of $0 \leq k' < k$ planted vectors and $k-k'$ non-planted vectors form a set of k independent p-biased vectors which is therefore a solution to the k-OV problem with probability $(1-p)^d = \frac{1}{n^{2k}}$. By linearity of expectation,

$$\mathbb{E}[c(\mathbf{U})] = 1 + (1-p^k)^d \cdot (n^k - 1) < 1 + \frac{1}{n^k}.$$

Which then gives the desired result.

$$\Pr[c(U) > 1] = \sum_{i=2}^{n^k} \Pr[c(U) = i] \leq \sum_{i=1}^{n^k} (i-1) \cdot \Pr[c(U) = i] = \mathbb{E}[c(\mathbf{U}) - 1] < \frac{1}{n^k}.$$

□

Uniqueness. Our way of planting is unique in the following sense.

Theorem 1. *Let $Q : \{0,1\}^k \to \mathbb{R}$ be a probability distribution such that $Q(1^k) = 0$ and that marginalizing any one of the k bits yields $k-1$ independent p-biased bits. Then $Q = \mathcal{P}_k$.*

Proof. We show that $Q(x) = \mathcal{P}_k(x)$ for all $x \in \{0,1\}^k$. Let m denote the number of ones in x, that is $||x||_1 = m$. We proceed by induction over $k-m$, i.e. the number of *zeros* in x.

Base case: $k - m = 0$. Then $m = k$ and $x = 1^k$. Thus $Q(x) = Q(1^k) = 0 = \mathcal{P}_k(x)$.

Inductive case: $k - m > 0$. We assume w.l.o.g. that the $k-m$ zeros are the *last* bits of x, i.e., $x = 1^m 0^{k-m}$. Marginalizing the final $k-m > 0$ bits of Q yields $k-(k-m) = m$ independent p-biased bits, whereby the probability of all m remaining bits being ones is

$$p^m = \sum_{y \in \{0,1\}^{k-m}} Q(1^m y) = Q(\underbrace{1^m 0^{m-k}}_{=x}) + \sum_{\substack{y \in \{0,1\}^{k-m} \\ y \neq 0^{k-m}}} Q(1^m y).$$

Thereby,

$$\begin{aligned} Q(x) &= p^m - \sum_{\substack{y \in \{0,1\}^{k-m} \\ y \neq 0^{k-m}}} Q(1^m y) \\ &= p^m - \sum_{\substack{y \in \{0,1\}^{k-m} \\ y \neq 0^{k-m}}} \mathcal{P}_k(1^m y) && \text{(By the induction hypothesis)} \\ &= p^m + \mathcal{P}_k(x) - \sum_{y \in \{0,1\}^{k-m}} \mathcal{P}_k(1^m y) \end{aligned}$$

where the sum term is merely the probability of 1^m in the marginal distribution of $\mathcal{P}_k$, which by Lemma 4 in turn consists of m independent p-biased bits. Hence

$$= p^m + \mathcal{P}_k(x) - p^m = \mathcal{P}_k(x).$$

□

5 Conjectured hard problems

In this section we formally define the problems that we conjecture to require $n^{k-o(1)}$ time.

Definition 1 (Solving planted **decision** k-OV)**.** *Let $\mathcal{A}$ be an algorithm that given a k-OV instance $\mathbf{U}$ outputs either 0 or 1. For $\alpha(n) = \Omega(1)$, we say $\mathcal{A}$* solves *the* decision *k-OV$^\alpha$ problem with success probability $\delta(n)$, if for both $b \in \{0,1\}$ and large enough n,*

$$\Pr_{\mathbf{U} \sim k\text{-}OV_b^\alpha(n)} [\mathcal{A}(\mathbf{U}) = b] \geq \delta(n),$$

where randomness is taken over both the instance $\mathbf{U}$ and the random coins used by $\mathcal{A}$.

Similarly, we define a notion of recovering a solution from a planted instance.

Definition 2 (Solving planted **search** k-OV)**.** *Let $\mathcal{A}$ be an algorithm that given a k-OV instance* $\mathbf{U}$ *outputs a tuple $(s_1, ..., s_k) \in \{1, ..., n\}^k$. For a given $\alpha(n) = \Omega(1)$, we say $\mathcal{A}$* solves *the* planted search k-OV$^\alpha$ problem *with success probability $\delta(n)$ if for large enough n,*

$$\Pr_{\substack{\mathbf{U} \sim k\text{-}OV_1^\alpha(n) \\ (s_1,...,s_k) \leftarrow \mathcal{A}(\mathbf{U})}} [U_{1,s_1}, ..., U_{k,s_k} \text{ are orthogonal}] \geq \delta(n),$$

where randomness is taken over both the instance $\mathbf{U}$ *and the random coins used by $\mathcal{A}$.*

Now we are ready to formally state our main conjecture.

Conjecture 1. *For any $\alpha(n) = \omega(1)$ and $\varepsilon > 0$, there exists no algorithm $\mathcal{A}$ that solves the planted decision k-OV problem with any constant success probability $\delta > \frac{1}{2}$ in time $O(n^{k-\varepsilon})$.*

6 Search-to-decision reduction via binary search

We reduce the search problem of finding the planted solution to the decision problem of determining whether an instance contains a planted solution. This means that given a decision algorithm that can correctly distinguish whether an instance was sampled from the model or planted distribution with sufficient probability, one can recover the planted secret through this reduction. The reduction introduces a factor $O(\log n)$ increase in both the running time and error probability of the algorithm.

The idea is to find each planted vector using something akin to binary search on each collection A_i. We can split A_i into two partitions of roughly equal size and run the decision algorithm twice, on instances where one of the two partitions is first replaced by newly sampled p-biased vectors. The vector planted in A_i is guaranteed to be replaced in one of these cases, and by $(k-1)$-wise independence the resulting instance follows the model distribution. The search space is thus cut in half and we can recurse on this smaller search space to eventually find the planted vector.

Theorem 2 (Search to decision reduction)**.** *Let $\alpha(n) = \text{polylog}(n)$ and let $\mathcal{A}^{\text{decide}}$ be an algorithm that solves the planted decision k-OV problem with success probability $1 - \delta(n)$ in time $T(n)$. Then there exists an algorithm $\mathcal{A}^{\text{search}}$ that solves the planted search k-OV problem with success probability $1 - k\lceil \log n \rceil \cdot \delta(n)$ in expected time $\widetilde{O}(T(n) + n)$.*

Proof. Consider an instance $\mathbf{U} = (U_1, \ldots, U_k) \sim k\text{-OV}_1^\alpha(n)$. First let us focus only on recovering the location $i \in [n]$ of the planted vector in the first collection U_1. The reduction begins with the "full" search space $S := [n]$, and narrows it down by half in each iteration, so that the desired i is recovered after $\lceil \log n \rceil$ iterations.

At each iteration, the current search space S is arbitrarily partitioned in two sets of equal size (up to one vector, id . The decision algorithm $\mathcal{A}^{\text{decide}}$ is then executed on two new instances, where the respective sets of vectors in U_1 are replaced with newly sampled p-biased vectors.

By the $(k-1)$-wise independence, if the vector belonging to the solution is replaced, all vectors are independently and identically distributed p-biased vectors, i.e., the instance is distributed according to $k\text{-OV}_0^\alpha(n)$. On the other hand, if the solution survives resampling, the instance remains distributed according to $k\text{-OV}_1^\alpha(n)$. Therefore, the output of $\mathcal{A}^{\text{decide}}$ is used to decide which of the two partition blocks should be assumed as the new search space.

The reduction is correct if $\mathcal{A}^{\text{decide}}$ decides correctly at every iteration. Of course, $\mathcal{A}^{\text{decide}}$ might fail with probability $\delta(n)$. By a union bound over all $\lceil \log n \rceil$ invocations of $\mathcal{A}^{\text{decide}}$ this happens with probability at most $\lceil \log n \rceil \cdot \delta(n)$. Thereby $\mathcal{A}^{\text{search}}$ recovers the location i of the first planted vector with success probability at least $1 - \lceil \log n \rceil \cdot \delta(n)$.

As for the runtime, $\mathcal{A}^{\text{decide}}$ with runtime $T(n)$ is invoked $O(\log n)$ times, and across all iterations $n-1$ vectors are resampled in total. Since a single p-biased bit can be sampled in expected time $O(-\log p) = O(\log \alpha(n)) = O(\log \log n)$, sampling a d-dimensional vector takes $\operatorname{polylog}(n)$ time in expectation. Therefore, recovering the location of the first planted vector takes time $\widetilde{O}(T(n) + n)$.

The same process is repeated another $k-1$ times to recover the locations of the planted vectors among $U_2, ..., U_k$. As k is constant there is no asymptotic overhead in the running time while the success probability drops to $1 - k\lceil \log n \rceil \cdot \delta(n)$. □

7 Search-to-decision reduction via counters

We present a second search-to-decision reduction, adapted from that of Agrawal et al. [2] for planted k-SUM. As in the method in Section 6, we use the fact that an algorithm $\mathcal{A}^{\text{decide}}$ for the decision k-OV problem, when given a planted k-OV instance with some of the vectors resampled, correctly detects whether any of the planted vectors were among the resampled vectors. However, instead of iteratively narrowing a pool of candidate vectors, we iterate this process on the entire instance and for each vector u keep a count of the number of iterations in which u survived and $\mathcal{A}^{\text{decide}}$'s output was 1. After $O(\log(n))$ iterations we output the vectors with the highest counts among all k collections, which we show coincides with the planted solution (with good probability).

Theorem 3 (Search to decision reduction)**.** *For any $\alpha(n) = \operatorname{polylog}(n)$ and $\delta > 0$, there exists an $\epsilon > 0$ such that an algorithm solves the planted decision k-OV problem with success probability at least $1-\epsilon$ in time $T(n)$ yields an algorithm that solves the planted search k-OV problem with success probability at least $1-\delta$ in expected time $\widetilde{O}(T(n)+n)$.*

In more detail, let Mix be the following randomized algorithm, which takes a k-OV instance $\mathbf{U}$, and resamples some of the vectors:

Algorithm $\mathsf{Mix}(\mathbf{U})$:

```
1. For each $\ell \in [k]$ and $i \in [n]$:
    (a) With probability $1 - 2^{-\frac{1}{k}}$, replace $U_{\ell,i}$ by a newly-sampled $p$-biased vector
2. Output $\mathbf{U}$
```

For $\ell \in [k]$, let $S_\ell \subseteq [n]$ indicate the indices of the vectors of U_ℓ which are replaced by Mix. For a vector u in $\mathbf{U}$ and a given execution of Mix, we say u *survives* if Mix does not replace u. We say $\mathbf{s} = (s_1, \ldots, s_k)$ *survives* if U_{ℓ,s_ℓ} survives for each $\ell \in [k]$.

Now let $B(\mathbf{s})$ be the binary random variable indicating whether $\mathbf{s}$ survives. Our chosen probability for Mix to resample a vector yields the following.

Lemma 6. *For a k-OV instance $\mathbf{U}$ and any $\mathbf{s} \in [n]^k$, $\mathbf{s}$ survives with probability one half, i.e., $\Pr_{\mathsf{Mix}}[B(\mathbf{s}) = 1] = \frac{1}{2}$.*

Proof. Each vector is independently picked to be replaced with probability $1 - 2^{-\frac{1}{k}}$. The chance of all k vectors surviving is

$$\Pr_{\mathsf{Mix}}[B(\mathbf{s}) = 1] = \left(1 - \left(1 - 2^{-\frac{1}{k}}\right)\right)^k = 2^{-\frac{1}{k}\cdot k} = \frac{1}{2}.$$

□

As laid out before, our search algorithm repeatedly executes Mix and then the given decision algorithm $\mathcal{A}^{\text{decide}}$. The hopes is that the output of the latter correlates with the survival of the planted solution. We therefore also keep track of which vectors survive whenever $\mathcal{A}^{\text{decide}}$ believes there is still a planted solution in the instance output by Mix.

Algorithm $\mathcal{A}^{\text{search}}(\mathbf{U})$:

```
1. Initialize a counter $C_{\ell,i} := 0$ for each $\ell \in [k]$ and $i \in [n]$.
2. Repeat $m = \Theta(\log n)$ times:
   (a) $\mathbf{V} \leftarrow \mathsf{Mix}(\mathbf{U})$
   (b) $b := \mathcal{A}^{\text{decide}}(\mathbf{V})$
   (c) If $b = 1$:
       i. Set $C_{\ell,i} := C_{\ell,i} + 1$ for every $U_{\ell,i}$ that was not replaced by $\mathsf{Mix}$
3. Set $s_\ell := \arg\max_{i \in [n]} C_{\ell,i}$ for each $\ell \in [k]$
4. Output $\mathbf{s} = (s_1, \ldots, s_k)$
```

This works well for instances $\mathbf{U}$ where $\mathcal{A}^{\text{decide}}$ is good at detecting whether a particular solution survives. To capture this notion, we say an instance $\mathbf{U}$ is *good*, if it has only one solution, at some location $\mathbf{s}$, *and* the output of $\mathcal{A}^{\text{decide}}$ indicates whether $\mathbf{s}$ survives except for a small constant probability; Concretely,

$$\Pr_{\mathsf{Mix}}[\mathcal{A}^{\text{decide}}(\mathsf{Mix}(\mathbf{U})) \neq B(\mathbf{s})] < \frac{1}{2^3}\left(1 - 2^{-\frac{1}{k}}\right).$$

In the following, let Sample be a randomized algorithm that outputs a planted instance sampled from k-$\mathrm{OV}_1^\alpha(n)$ as well as the location $\mathbf{s}$ of the planted solution.

Lemma 7. *For any $\delta > 0$, there exists an $\epsilon > 0$ such that given an algorithm that solves the planted decision k-OV problem with success probability at least $1 - \epsilon$, an instance $\mathbf{U} \sim k$-$OV_1^\alpha(n)$ is* good *except with probability at most $\frac{\delta}{2}$.*

Proof. We show first that for an apt choice of ϵ, the decision algorithm $\mathcal{A}^{\text{decide}}$ correctly detects if the solution planted by Sample (which produces instances distributed according to k-$\mathrm{OV}_1^\alpha(n)$) survives except with probability at most $\frac{\delta}{4}$. Then, since for large enough n, instances $\mathbf{U} \sim k$-$\mathrm{OV}_1^\alpha(n)$ will only contain a single solution except with probability $\frac{\delta}{4}$, by a union bound we get a good instance except with probability at most $\frac{\delta}{2}$.

For the first step, consider the following distribution:

Distribution k-$\mathrm{OV}_B^\alpha(n)$:

1. $\mathbf{U}, \mathbf{s} \leftarrow \mathsf{Sample}$
2. $\mathbf{V} \leftarrow \mathsf{Mix}(\mathbf{U})$
3. Output $(\mathbf{V}, B = B(\mathbf{s}))$.

Observe that, if we condition on $B = 1$, the planted solution survives Mix, which merely resamples some of the i.i.d. p-biased vectors in the instance. Thus, $\mathbf{V}$ is distributed according to $k\text{-OV}_1^\alpha(n)$. On the other hand, if $B = 0$, at least one of the planted vectors is replaced by a newly-sampled p-biased vector. By $(k-1)$-wise independence, the subset of planted vectors that survive are i.i.d. p-biased vectors, as are all other vectors in $\mathbf{V}$. Hence, conditioning on $B = 0$ yields the model distribution $k\text{-OV}_0^\alpha(n)$.

Therefore, for an algorithm $\mathcal{A}^{\text{decide}}$ that solves the planted decision k-OV problem with success probability at least $1 - \epsilon$, we have

$$\Pr_{k\text{-OV}_B^\alpha(n)}[\mathcal{A}^{\text{decide}}(\mathbf{V}) \neq B] \leq \epsilon.$$

Now, let $Z(\mathbf{U}, \mathbf{s})$ be the random variable that, for a given instance $\mathbf{U}$ with a solution planted at $\mathbf{s}$, denotes the probability of $\mathcal{A}^{\text{decide}}(\mathsf{Mix}(U)) \neq B(\mathbf{s})$, where the randomness is taken over the internal coins used by Mix. Then

$$\begin{aligned} \epsilon &\geq \Pr_{k\text{-OV}_B^\alpha(n)}[\mathcal{A}^{\text{decide}}(\mathbf{V}) \neq B(\mathbf{s})] \\ &= \mathop{\mathbb{E}}_{(\mathbf{U},\mathbf{s})\leftarrow\mathsf{Sample}}\left[\Pr_{\mathsf{Mix}}[\mathcal{A}^{\text{decide}}(\mathsf{Mix}(\mathbf{U})) \neq B(\mathbf{s})]\right] \\ &= \mathop{\mathbb{E}}_{(\mathbf{U},\mathbf{s})\leftarrow\mathsf{Sample}}\left[Z(\mathbf{U}, \mathbf{s})\right]. \end{aligned}$$

Choosing $\epsilon < \frac{\delta}{2^5}(1 - 2^{-\frac{1}{k}})$, Markov's inequality yields the following bound:

$$\Pr_{(\mathbf{U},\mathbf{s})\leftarrow\mathsf{Sample}}\left[Z(\mathbf{U}, \mathbf{s}) \geq \frac{1}{2^3}\left(1 - 2^{-\frac{1}{k}}\right)\right] \leq \Pr_{(\mathbf{U},\mathbf{s})\leftarrow\mathsf{Sample}}\left[Z(\mathbf{U}, \mathbf{s}) \geq \frac{4}{\delta}\,\mathbb{E}[Z(\mathbf{U}, \mathbf{s})]\right] < \frac{\delta}{4}.$$

Next, observe that $\mathbf{s}$ is the only solution in the instance $\mathbf{U}$ output by Sample with high probability,

$$\Pr_{(\mathbf{U},\mathbf{s})\leftarrow\mathsf{Sample}}[\mathbf{U} \text{ has a solution besides } \mathbf{s}] = \Pr_{\mathbf{U}\leftarrow k\text{-OV}_1^\alpha(n)}[c(\mathbf{U}) > 1] < \frac{1}{n^k}.$$

In particular, for large enough n, there is more than one solution with probability less than $\frac{\delta}{4}$. Thus, by a union bound over this and our result in the first step, we find that an instance $\mathbf{U} \sim k\text{-OV}_1^\alpha(n)$ is good except with probability at most $\frac{\delta}{2}$:

$$\begin{aligned} &\Pr_{\mathbf{U}\leftarrow k\text{-OV}_1^\alpha(n)}[\mathbf{U} \text{ is good}] \\ &= \Pr_{\mathbf{U}\leftarrow k\text{-OV}_1^\alpha(n)}\left[\mathbf{U} \text{ has a single solution } \mathbf{s} \text{ and } Z(\mathbf{U}, \mathbf{s}) < \frac{1}{2^3}\left(1 - 2^{-\frac{1}{k}}\right)\right] \\ &= \Pr_{\mathbf{U},\mathbf{s}\leftarrow\mathsf{Sample}}\left[\mathbf{s} \text{ is the only solution of } \mathbf{U} \text{ and } Z(\mathbf{U}, \mathbf{s}) < \frac{1}{2^3}\left(1 - 2^{-\frac{1}{k}}\right)\right] \\ &\geq 1 - \Pr_{\mathbf{U},\mathbf{s}\leftarrow\mathsf{Sample}}[\mathbf{U} \text{ has a solution besides } \mathbf{s}] - \Pr_{\mathbf{U},\mathbf{s}\leftarrow\mathsf{Sample}}\left[Z(\mathbf{U}, \mathbf{s}) \geq \frac{1}{2^3}\left(1 - 2^{-\frac{1}{k}}\right)\right] \\ &> 1 - \frac{\delta}{4} - \frac{\delta}{4} = 1 - \frac{\delta}{2}. \end{aligned}$$

□

We now show that the search algorithm performs well on this large fraction of good instances.

Lemma 8. *Let $\delta > 0$ and let $\mathcal{A}^{\text{decide}}$ be an algorithm that solves the planted decision k-OV problem. Then $\mathcal{A}^{\text{search}}$ fails to recover the solution $\mathbf{s}$ of good instances with probability less than $\frac{\delta}{2}$.*

Proof. Let $\mathbf{U}$ be a good instance with its only solution at $\mathbf{s}$. After t iterations, we expect the counters for vectors in the solution $\mathbf{s}$ to be the highest:

$$\begin{aligned}
\mathbb{E}[C_{\ell,i} \mid s_\ell \neq i] &= t \cdot \Pr_{\mathsf{Mix}}\left[U_{\ell,i} \text{ survives and } \mathcal{A}^{\text{decide}}(\mathsf{Mix}(\mathbf{U})) = 1\right] \\
&\leq t \cdot \left(\Pr_{\mathsf{Mix}}\left[U_{\ell,i} \text{ survives and } B(\mathbf{s}) = 1\right] + \Pr_{\mathsf{Mix}}\left[\mathcal{A}^{\text{decide}}(\mathsf{Mix}(\mathbf{U})) \neq B(\mathbf{s})\right]\right) \\
&< t \cdot \left(\frac{1}{\sqrt[k]{2}} \cdot \frac{1}{2} + \frac{1}{2^3}\left(1 - \frac{1}{\sqrt[k]{2}}\right)\right),
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}[C_{\ell,i} \mid s_\ell = i] &\geq t \cdot \Pr_{\mathsf{Mix}}\left[\mathbf{s} \text{ survives and } \mathcal{A}^{\text{decide}}(\mathsf{Mix}(\mathbf{U})) = 1\right] \\
&\geq t \cdot \Pr_{\mathsf{Mix}}\left[B(\mathbf{s}) = 1 \text{ and } \mathcal{A}^{\text{decide}}(\mathsf{Mix}(\mathbf{U})) = B(\mathbf{s})\right] \\
&\geq t \cdot \left(1 - \Pr_{\mathsf{Mix}}\left[B(\mathbf{s}) = 0\right] - \Pr_{\mathsf{Mix}}\left[\mathcal{A}^{\text{decide}}(\mathsf{Mix}(\mathbf{U})) \neq B(\mathbf{s})\right]\right) \\
&> t \cdot \left(1 - \frac{1}{2} - \frac{1}{2^3}\left(1 - \frac{1}{\sqrt[k]{2}}\right)\right) = t \cdot \left(\frac{1}{2} - \frac{1}{2^3}\left(1 - \frac{1}{\sqrt[k]{2}}\right)\right).
\end{aligned}$$

Picking the k highest counters is guaranteed to yield the solution $\mathbf{s}$ if all counters $C_{\ell,i}$ deviate by less than half the difference of the two expected values, namely

$$\begin{aligned}
\Delta &:= \frac{1}{2}\left(\mathbb{E}[C_{\ell,i} \mid s_\ell = i] - \mathbb{E}[C_{\ell,i} \mid s_\ell \neq i]\right) \\
&\geq \frac{1}{2}t\left(\left(\frac{1}{2} - \frac{1}{2^3}\left(1 - \frac{1}{\sqrt[k]{2}}\right)\right) - \left(\frac{1}{\sqrt[k]{2}} \cdot \frac{1}{2} + \frac{1}{2^3}\left(1 - \frac{1}{\sqrt[k]{2}}\right)\right)\right). \\
&= \frac{1}{2}t\left(\frac{1}{2}\left(1 - \frac{1}{\sqrt[k]{2}}\right) - \frac{1}{2^2}\left(1 - \frac{1}{\sqrt[k]{2}}\right)\right) \\
&= t \cdot \frac{1}{8}\left(1 - \frac{1}{\sqrt[k]{2}}\right)
\end{aligned}$$

Each counter $C_{\ell,i}$ is the sum of t i.i.d. binary random variables. By a Chernoff bound, a counter $C_{\ell,i}$ for a vector which is not in the planted solution, i.e. $s_\ell \neq i$ will exceed its expected value by Δ with probability at most

$$\Pr\left[C_{\ell,i} \geq \mathbb{E}[C_{\ell,i}] + \Delta \mid s_\ell \neq i\right] < \exp\left(-\frac{2}{t}\left[\frac{1}{8}\left(1 - \frac{1}{\sqrt[k]{2}}\right)\right]^2\right).$$

Similarly, a counter $C_{\ell,i}$ for a vector that *is* part of a planted solution will fall short of its expected value by Δ with probability at most

$$\Pr\left[C_{\ell,i} \leq \mathbb{E}[C_{\ell,i}] - \Delta \mid s_\ell = i\right] < \exp\left(-\frac{2}{t}\left[\frac{1}{8}\left(1 - \frac{1}{\sqrt[k]{2}}\right)\right]^2\right).$$

For a choice of $t = \frac{1}{2} \cdot \left[\frac{1}{8}\left(1 - \frac{1}{\sqrt[k]{2}}\right)\right]^{-2} \log \frac{2kn}{\delta} = O(\log n)$ iterations, both of these probabilities are at most $\frac{\delta}{2kn}$. If *none* of the $k \cdot n$ counters deviate by Δ, the ranges of counters for vectors at $\mathbf{s}$ and those for vectors not at $\mathbf{s}$ will be disjoint and selecting for the highest counters is guaranteed to yield $\mathbf{s}$. Thus, by a union bound over all $k \cdot n$ counters, we fail to recover $\mathbf{s}$ with probability at most $kn \cdot \frac{\delta}{2kn} = \frac{\delta}{2}$. □

A runtime analysis of $\mathcal{A}^{\text{search}}$ now completes the proof of our theorem.

Proof of Theorem 3. By Lemma 7, there exists an $\epsilon > 0$ such that for a given algorithm $\mathcal{A}^{\text{decide}}$ that solves the planted decision k-OV problem with success probability at least $1 - \epsilon$, an instance $\mathbf{U} \sim k\text{-OV}_1^{\alpha}(n)$ is good except with probability at most $\frac{\delta}{2}$. By Lemma 8, $\mathcal{A}^{\text{search}}$ is able to recover the solution $\mathbf{s}$ from a good instance except with probability at most $\frac{\delta}{2}$. By a union bound against the instance not being good or $\mathcal{A}^{\text{search}}$ failing on the good instance, $\mathcal{A}^{\text{search}}$ solved the planted search k-OV problem with success probability at least $1 - \delta$.

As laid out in the proof of Theorem 2, a single d-dimensional p-biased vector can be sampled polylog(n) expected time. On average, a single execution of Mix samples $k \cdot n \cdot (1 - 2^{-\frac{1}{k}}) = O(n)$ new vectors, which therefore can be done in $\widetilde{O}(n)$ time.

As for $\mathcal{A}^{\text{search}}$, both the initialization and the final evaluation of the counters takes linear time; The $t = O(\log n)$ iterations each involve one execution of Mix and $\mathcal{A}^{\text{decide}}$, which take $\widetilde{O}(n)$ and $T(n)$ time respectively for an overall runtime of $\widetilde{O}(T(n) + n)$. □

Acknowledgements. We are grateful to Andrej Bogdanov, Antoine Joux, Moni Naor, Nikolaj Schwartzbach, and Prashant Vasudevan for insightful discussions.

References

[1] Amir Abboud, Richard Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 218–230. SIAM, 2015. doi:10.1137/1.9781611973730.17.

[2] Shweta Agrawal, Sagnik Saha, Nikolaj I. Schwartzbach, Akhil Vanukuri, and Prashant Nalini Vasudevan. k-SUM in the sparse regime: Complexity and applications. In *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference*, volume 14921 of *Lecture Notes in Computer Science*, pages 315–351. Springer, 2024. doi:10.1007/978-3-031-68379-4_10.

[3] Josh Alman, Alexandr Andoni, and Hengjie Zhang. Faster algorithms for average-case orthogonal vectors and closest pair problems. In *2025 Symposium on Simplicity in Algorithms (SOSA)*, pages 473–484. SIAM, 2025. doi:10.1137/1.9781611978315.35.

[4] Josh Alman, Yizhi Huang, and Kevin Yeo. Fine-grained complexity in a world without cryptography. *IACR ePrint*, page 324, 2025. URL: https://eprint.iacr.org/2025/324.

[5] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 483–496, 2017.

[6] Timothy M Chan and R Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky. *ACM Transactions on Algorithms (TALG)*, 17(1):1–14, 2020.

[7] Mina Dalirrooyfard, Andrea Lincoln, Barna Saha, and Virginia Vassilevska Williams. Average-case hardness of parity problems: Orthogonal vectors, k-sum and more. *CoRR*, abs/2503.21951, 2025. URL: `https://doi.org/10.48550/arXiv.2503.21951`.

[8] Mina Dalirrooyfard, Andrea Lincoln, and Virginia Vassilevska Williams. New techniques for proving fine-grained average-case hardness. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 774–785, 2020.

[9] Itai Dinur, Nathan Keller, and Ohad Klein. Fine-grained cryptanalysis: Tight conditional bounds for dense k-SUM and k-XOR. *J. ACM*, 71(3):23, 2024. Announced at FOCS 2021. `doi:10.1145/3653014`.

[10] Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and Ryan Williams. Completeness for first-order properties on sparse structures with algorithmic applications. *ACM Transactions on Algorithms (TALG)*, 15(2):1–35, 2018.

[11] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. Announced at FOCS 1998. `doi:10.1006/JCSS.2001.1774`.

[12] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 44–61. ACM, 1989. `doi:10.1145/73007.73012`.

[13] Ari Juels and Marcus Peinado. Hiding cliques for cryptographic security. *Designs, Codes and Cryptography*, 20(3):269–280, 2000.

[14] Daniel M. Kane and Richard Ryan Williams. The orthogonal vectors conjecture for branching programs and formulas. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019*, volume 124 of *LIPIcs*, pages 48:1–48:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/LIPICS.ITCS.2019.48`.

[15] Rio LaVigne, Andrea Lincoln, and Virginia Vassilevska Williams. Public-key cryptography in the fine-grained setting. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference*, volume 11694 of *Lecture Notes in Computer Science*, pages 605–635. Springer, 2019. `doi:10.1007/978-3-030-26954-8\_20`.

[16] Ralph C. Merkle. Secure communications over insecure channels. *Commun. ACM*, 21(4):294–299, 1978. `doi:10.1145/359460.359473`.

[17] Alon Rosen. Fine-grained cryptography: A new frontier? *IACR Cryptol. ePrint Arch.*, page 442, 2020. URL: `https://eprint.iacr.org/2020/442`.

[18] Virginia Vassilevska Williams. *On some fine-grained questions in algorithms and complexity*, pages 3447–3487. World Scientific, 2018. `doi:10.1142/9789813272880_0188`.

[19] R. Ryan Williams. Some estimated likelihoods for computational complexity. In *Computing and Software Science - State of the Art and Perspectives*, volume 10000 of *Lecture Notes in Computer Science*, pages 9–26. Springer, 2019. `doi:10.1007/978-3-319-91908-9\_2`.

[20] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. `doi:10.1016/J.TCS.2005.09.023`.

A Planting more solutions

So far we have only considered planting a single solution in an instance sampled from the model distribution. We justified our particular planted distribution by the fact it is the only way of sampling solutions (orthogonal combinations of vectors) such that every subset of $k' < k$ vectors is distributed identically to k' independent p-biased vectors. As a consequence, if we have multiple sets of orthogonal vectors sampled this way, any combination of k vectors that are not all part of the same set are distributed identically to k independent p-biased vectors.

We propose a family of distribution, called *m-planted-k-*$\mathrm{OV}_0^\alpha(n)$, for $m \in [n]$, where m sets of k orthogonal vectors are sampled independently and embedded at m (disjoint) locations in a k-$\mathrm{OV}_0^\alpha(n)$ instance. This is a generalization of the previous planted distribution which corresponds to the case of $m = 1$. On the other extreme, when $m = n$, all $k \cdot n$ vectors in an m-planted-k-$\mathrm{OV}_0^\alpha(n)$ instance belong to some orthogonal combination and no vector is an independently sampled p-biased vector. The task of finding a single orthogonal combination of vectors is therefore trivially easier in this case than in the case of a 1-planted k-$\mathrm{OV}_0^\alpha(n)$ instance, as one can simply pick any one vector arbitrarily, and find $k-1$ vectors from the other collections to form an orthogonal combination through exhaustive search in time $O(n^{k-1}d) = o(n^k)$. Recovering the entire solution, that is all n many k-tuples of locations where the orthogonal combinations were planted, still takes time $\Theta(n^k d)$ when done in the naive way.

In general, a solution to the m-planted k-$\mathrm{OV}_0^\alpha(n)$ problem can be encoded as k sets $S_1, ..., S_k \subseteq \{1, ..., n\}$ of size m, indicating which vectors within collection $U_1, ..., U_k$ are part of an orthogonal combination, as well as $k-1$ permutations $\pi_2, ..., \pi_k \in S_m$ that encode how these vectors match up to form orthogonal combinations. We call $S_1, ..., S_k$ the *support of the solution* and $\pi_2, ..., \pi_k$ the *matching*. There are therefore

$$\binom{n}{m}^k \cdot (m!)^{k-1}$$

possible solutions and the information encoded in an instance amounts to

$$\begin{aligned}\log\left(\binom{n}{m}^k \cdot (m!)^{k-1}\right) &= k \log \binom{n}{m} + (k-1)\log(m!) \\ &= O(m \log n)\end{aligned}$$

As an observation, in the case of 1-planted k-$\mathrm{OV}_0^\alpha(n)$, the matching is trivial (as there is only one planted orthogonal combination), and therefore the difficulty in the problem is derived from finding the support. On the other hand, for n-planted k-$\mathrm{OV}_0^\alpha(n)$, the support is trivial (as $S_1 = ... = S_k = \{1, ..., n\}$) and the difficulty lies solely in determining the matching.

B The downsampling attack

In this section we give another argument – different than the algorithms by Abboud et al. [1] and Chan and Williams [6], and simpler than them – for choosing $d = \omega(\log n)$. We call it the *downsampling attack*. It relies on the fact k-OV can be solved faster than the n^k barrier in the sublogarithmic dimension regime $d = o(\log n)$. In this regime, by the pigeonhole principle there are necessarily duplicate vectors in the instance, and it becomes faster to enumerate all possible solutions and check if (or how often) they occur.

Now, take a higher dimensional instance of planted k-OV, and project it onto the first $d = o(\log n)$ coordinates – the resulting instance still contains the original planted solution, but now it might not be unique as we expect random solutions to occur among the p-biased vectors for such a low dimension. However, through enumeration one can find all solutions among the smaller instance, which act as candidate solutions for the original instance. Unless almost all combinations of vectors in the smaller instance are orthogonal, this attack can find and enumerate the candidate solutions in $O(n^{k-\epsilon})$ time and then check whether any of these candidates is a solution to the original instance.

More formally, let $k \geq 2$, $\alpha(n) = \Omega(1)$, L be an instance of planted k-OV$_c(n)$, and $\epsilon > 0$. Pick $\delta := \epsilon\frac{\alpha(n)}{2k} < 1 - \epsilon$ (for small enough ϵ), and create a copy L' of L projected onto the first $d' = \delta \log_2 n$ coordinates.

We can then enumerate all $O(2^{k\times d'})$ combinations of k d'-dimensional orthogonal vectors and find all their occurrences in L' in $\tilde{O}(2^{kd'} + n) = \tilde{O}(n^{k\cdot\delta} + n) = \tilde{O}(n^{k-\epsilon})$ time. On expectation there will be $(1-p^k)^{d'} \cdot n^k = 2^{-\frac{2k}{c(n)}\cdot d'} \cdot n^k = n^{k-\epsilon}$ such candidate combinations, which can then be tested in the full instance L in $\tilde{O}(n^{k-\epsilon})$ time. The algorithm is correct, as the planted solution in L is guaranteed to also be a solution in L'.

Therefore, if $\alpha(n) = \Theta(1)$, we can find an $\epsilon > 0$ such that this attack finds the planted solution in $O(n^{k-\epsilon})$ expected time. If α is super-constant however, i.e. $\alpha = \omega(1)$, then the advantage ϵ (subject to $\epsilon\frac{\alpha(n)}{2k} < 1 - \epsilon$) tends towards zero asymptotically. Hence, $\alpha(n) = \omega(1)$ is a necessary condition for the instances to be $n^{k-o(1)}$-hard.