3SUM with Preprocessing: Algorithms, Lower Bounds and Cryptographic Applications

Alexander Golovnev
Harvard
alexgolovnev@gmail.com

Siyao Guo
NYU Shanghai
sg191@nyu.edu

Thibaut Horel
Harvard
thorel@seas.harvard.edu

Sunoo Park
MIT & Harvard
sunoo@mit.edu

Vinod Vaikuntanathan
MIT
vinodv@csail.mit.edu

Abstract

Given a set of integers $\{a_1, \dots, a_N\}$, the 3SUM problem requires finding $a_i, a_j, a_k \in A$ such that $a_i + a_j = a_k$. A preprocessing version of 3SUM, called 3SUM-Indexing, considers an initial offline phase where a computationally unbounded algorithm receives $a_1, \dots, a_N$ and produces a data structure with S words of w bits each, followed by an online phase where one is given the target b and needs to find a pair (i, j) such that $a_i + a_j = b$ by probing only T memory cells of the data structure. In this paper, we study the 3SUM-Indexing problem and show the following.

New algorithms: 3SUM-Indexing with preprocessing has a simple algorithm with $S = O(N)$ and $T = O(N)$, and another with $S = O(N^2)$ and $T = \widetilde{O}(1)$. Goldstein et al. conjectured that this is the best possible, more precisely that there is no data structure with $S = N^{2-\varepsilon}$ and $T = N^{1-\varepsilon}$ for any constant $\varepsilon > 0$. Our first contribution is to disprove this conjecture by showing a suite of algorithms with $S^3 \cdot T = \widetilde{O}(N^6)$; for example, this achieves $S = \widetilde{O}(N^{1.9})$ and $T = \widetilde{O}(N^{0.3})$.

New lower bounds: Demaine and Vadhan in 2001 showed that every 1-query algorithm for 3SUM-Indexing requires space $\widetilde{\Omega}(N^2)$. Our second result generalizes their bound to show that for every space-S algorithm that makes T non-adaptive queries, $S = \widetilde{\Omega}(N^{1+1/T})$, giving us non-trivial statements for $T = o(\log N)$. Any asymptotic improvement to our result will result in a major breakthrough in static data structure lower bounds.

New cryptographic applications: "Backdoors" in cryptographic algorithms are a grave concern, especially following recent revelations of weaknesses found in certain standardized cryptographic primitives. A natural question is whether one can *modify* an entropic but imperfect (unkeyed) function, which a powerful adversary may have tampered with, into a function which is provably hard to invert even to such an adversary. That is, can we use a "backdoored" random oracle to build secure cryptography? We provide a novel formulation of this problem, modeling a random oracle whose truth table can be arbitrarily preprocessed by an unbounded adversary into an exponentially large lookup table to which the online adversary has oracle access. We construct one-way functions in this model assuming the hardness of a natural average-case variant of 3SUM-Indexing.

1 Introduction

One of the many equivalent formulations of the 3SUM problem is the following: given a set A of N integers, output $a_1, a_2, a_3 \in A$ such that $a_1 + a_2 = a_3$. There is an easy $O(N^2)$ time deterministic algorithm for 3SUM. Conversely, the popular 3SUM conjecture states that there are no sub-quadratic algorithms for this problem [GO95, Eri99].

Conjecture 1 (The "Modern 3SUM conjecture")**.** *There is no algorithm for 3SUM running in time $O(N^{2-\delta})$ for some constant $\delta > 0$.*

In this paper, we focus on a preprocessing variant of 3SUM known as 3SUM-Indexing, which was first defined by Demaine and Vadhan [DV01] in an unpublished note and then by Goldstein, Kopelowitz, Lewenstein and Porat [GKLP17]. In 3SUM-Indexing, there is an offline phase where a computationally *unbounded* algorithm receives $A = \{a_1, \ldots, a_N\}$ and produces a data structure with m words of w bits each; and an online phase which is given the target b and needs to find a pair (a_1, a_2) such that $a_1 + a_2 = b$ by probing only T memory cells of the data structure (*i.e.*, taking "query time" T). Note that the online phase does not receive the set A directly.

There are two simple algorithms that solve 3SUM-Indexing. The first stores a sorted version of A as the data structure (so $S = N$) and in the online phase, solves 3SUM-Indexing in $T = O(N)$ time using the standard two-finger algorithm for 3SUM. The second stores all pairwise sums of A, sorted, as the data structure (so $S = O(N^2)$) and in the online phase, looks up the target b in $T = \widetilde{O}(1)$ time.[1] There were no other algorithms known prior to this work. This led [DV01, GKLP17] to formulate the following three conjectures.

Conjecture 2 ([GKLP17])**.** *If there exists an algorithm which solves 3SUM-Indexing with preprocessing space S and $T = \widetilde{O}(1)$ probes then $S = \widetilde{\Omega}(N^2)$.*

Conjecture 3 ([DV01])**.** *If there exists an algorithm which solves 3SUM-Indexing with preprocessing space S and T probes, then $ST = \widetilde{\Omega}(N^2)$.*

Conjecture 4 ([GKLP17])**.** *If there exists an algorithm which solves 3SUM-Indexing with $T = \widetilde{O}(N^{1-\delta})$ probes for some $\delta > 0$ then $S = \widetilde{\Omega}(N^2)$.*

Note that these conjectures are in ascending order of strength:

$$\text{Conjecture 4} \Rightarrow \text{Conjecture 3} \Rightarrow \text{Conjecture 2}.$$

In terms of lower bounds, Demaine and Vadhan [DV01] showed that any 1-probe data structure for 3SUM-Indexing requires space $S = \widetilde{\Omega}(N^2)$. They leave the case of $T > 1$ open. Goldstein et al. [GKLP17] established connections between Conjectures 2 and 4 and hardness of Set Disjointness, Set Intersection, Histogram Indexing and Forbidden Pattern Document Retrieval.

1.1 Our contributions

Our contributions are three-fold. First, we show better algorithms for 3SUM-Indexing, refuting Conjecture 4. Second, we improve the lower bound of [DV01] to arbitrary T; our lower bound gives a non-trivial space bound for $T = o(\log N)$ non-adaptive queries. As we argue later, any asymptotic improvement to our lower bound will result in a major breakthrough in static data structure lower bounds. Third and finally, we show how to use the conjectured hardness of 3SUM-Indexing to enable a new cryptographic application, namely that of removing backdoors from unkeyed cryptographic functions. Next, we give a brief overview of these results in turn.

[1] The notation $\widetilde{O}(f(N))$ suppresses poly-logarithmic factors in $f(N)$.

1.1.1 Upper bound for 3SUM-Indexing

Theorem 1. *For every* $0 \leq \delta \leq 1$*, there is an* adaptive *data structure for 3SUM-Indexing with space* $S = \widetilde{O}(N^{2-\delta})$ *and query time* $T = \widetilde{O}(N^{3\delta})$*.*

In particular, Theorem 1 implies that by taking $\delta = 0.1$, we get a data structure which solves 3SUM-Indexing in space $S = \widetilde{O}(N^{1.9})$ and $T = \widetilde{O}(N^{0.3})$ probes, and, thus, refutes Conjecture 4.

In a nutshell, the upper bound starts by considering the function $f(i, j) = a_i + a_j$. This function has a domain of size N^2 but a potentially much larger range. In a preprocessing step, we convert this into a function g with a range of size $O(N^2)$ as well, such that inverting g lets us invert f. Once we have such a function, we use a result of Fiat and Naor [FN00] who give a general space-time tradeoff for inverting functions. This result gives non-trivial data structures for function *inversion* as long as function *evaluation* can be done efficiently. Due to our definitions of the functions f and g, we can efficiently compute them at every input, which leads to efficient inversion of f, and, therefore, efficient solution to 3SUM-Indexing. For more details, see Section 4. We note that prior to this work the result of Fiat and Naor [FN00] was recently used by Corrigan-Gibbs and Kogan [CGK18] for other algorithmic and complexity applications. In a concurrent work, Kopelowitz and Porat [KP19] obtain a similar upper bound for 3SUM-Indexing.

1.1.2 Lower bound for 3SUM-Indexing

We show that any algorithm for 3SUM-Indexing that uses a small number of probes requires large space, as expressed formally in Theorem 2.

Theorem 2. *For every* non-adaptive *algorithm that uses space* S *and query time* T *and solves 3SUM-Indexing, it holds that* $S = \widetilde{\Omega}(N^{1+1/T})$*.*

The lower bound gives us meaningful (super-linear) space bounds for nearly logarithmic T. Showing super-linear space bounds for static data structures for $T = \omega(\log N)$ probes is a major open question with significant implications [Sie04, Păt11, PTW10, Lar12, DGW19]. Essentially the only known technique for proving super-linear space lower bounds for $T = O(\log N)$ is cell-sampling. While the standard cell-sampling argument does not apply to the 3SUM-Indexing problem since this problem does not have the expansion property under any distribution, we are able to recover the corresponding lower bound for 3SUM-Indexing via a related method in the *non-adaptive* case.

In a nutshell, our lower bound proceeds by an incompressibility argument (introduced by Gennaro and Trevisan in [GT00], and later developed in [DTT10, DGK17]). That is, we show that any data structure with "surprising" space-time behavior can be used to compress a random set $A \subseteq [N^3]$ beyond its entropy. We refer the reader to Section 5 for more details on the proof.

1.1.3 Cryptographic application: "backdoored" random oracles

The power and prevalence of backdoors in cryptographic algorithms is a renewed concern given recent revelations [CMG+16, CNE+14, Gre13]. We consider a new model of a random oracle "backdoored" by a powerful adversary during a preprocessing phase.[2]

Recent results [Unr07, DGK17, CDGS18] studied the auxiliary-input random-oracle model in which an attacker can compute arbitrary S bits of auxiliary information about the function table of the random oracle H in a pre-processing phase, and make T additional queries to the random oracle in an online phase (where, for example, the adversary receives $y = H(x)$ that she wants to

[2] A random oracle $H : \{0,1\}^n \to \{0,1\}^m$ is simply a uniformly random function to which all parties have oracle access. We will let $N = 2^n$ throughout.

invert). However, it is easy to see that such a result cannot be true when $S \geq n \cdot 2^n$ since the preprocessed oracle can simply be the function table of the inverse function H^{-1}. This allows an adversary to invert H by making a single query to the S bits of preprocessed backdoor. In reality, we do wish to capture the possibility that the backdoor is the inverse function, which puts us in a conundrum.

A way out is to try to "immunize" the random oracle by designing a function $g(z) = g^H(z)$ which we wish to show is one-way even against an adversary that has oracle access to the backdoor for H. Let $g : \{0,1\}^{n'} \to \{0,1\}^{n'}$. To prevent the inverse table attack above, we require at the minimum that the size of the preprocessed table S is less than $n' \cdot 2^{n'}$.

This leads us to a natural static data structure problem with pre-processing. Imagine an adversary that can preprocess the table of H into S bits. Our goal is to construct a function $g(z) = g^H(z)$ (which necessarily acts on more than $\log S$ bits) which is one-way against an adversary who can make $T = \mathsf{poly}(n)$ queries to the S bits of preprocessed random oracle. Assuming the hardness of 3SUM with preprocessing for $T = \mathsf{poly}(n) = \mathsf{poly}(\log N)$ and space S, we show precisely such an immunization strategy. We refer the reader to Section 6 for more details on the construction.

We note the recent work of [BFM18] which circumvents the inverse table barrier in a different way, by assuming the existence of at least two independent (backdoored) random oracles. This allows them to use techniques from two-source extraction and communication complexity to come up with an immunization strategy.

2 Related work

2.1 Preprocessing attacks in cryptography

The power and limitation of preprocessing attacks have been studied in several contexts of cryptography, including time-space tradeoffs, non-uniform security and immunizing backdoors.

Time-space tradeoffs Hellman studied time-space tradeoffs for inverting random functions in his seminal paper [Hel80]. He showed that a random n-to-n bit function can be inverted with $2^{2n/3}$ online time and precomputed information. A series of followup works [FN00, BBS06, DTT10] studied time-space tradeoffs for inverting arbitrary one-way permutations, one-way functions and pseudorandom generators. In particular, Fiat and Naor [FN00] showed a $2^{3n/4}$ tradeoff for inverting any n-to-n bit function on every point.

Random oracles and non-uniform security Motivated by assessing the non-uniform security of hash functions, recent works [Unr07, DGK17, CDGS18] studied the auxiliary-input random-oracle model in which an attacker can compute arbitrary S bits of leakage before attacking the system and make T additional queries to the random oracle. Although our model is similar in that it allows preprocessed leakage of a random oracle, we differ significantly in two ways: the size of the leakage is larger, and the attacker only has oracle access to the leakage.

Specifically, their results and technical tools only apply to the setting where the leakage is smaller than the value table of the random oracle, whereas our model deals with leakage that is allowed to be larger. Furthermore, the random oracle model with auxiliary input allows the online adversary to access and depend on the leakage in an arbitrary way while our model only allows bounded number of oracle queries to the leakage, which is a more realistic model for online adversaries with bounded time and which cannot read the entire leakage at query time.

Immunizing backdoors and kleptography Motivated by immunizing backdoors, a series of recent works [DGG+15, BFM18, RTYZ18, FJM18] studied backdoored primitives including pseudorandom generators and hash functions. In this setting, the attacker might be given some space-bounded backdoor about a primitive, which could allow him to break the system more easily.

In particular, backdoored hash functions and random oracles are studied in [BFM18, FJM18]. Both of them observe that immunizing against a backdoor for a single unkeyed hash function might be hard. For this reason, [BFM18] considers the problem of combining two random oracles (with two independent backdoors). Instead, we look at the case of a single random oracle but add a restriction on the size of the advice. [FJM18] considers the setting of keyed functions such as (weak) pseudorandom functions, which are easier to immunize than unkeyed functions of the type we consider in this work.

The study of backdoored primitives is also related to — and sometimes falls within the field of — *kleptography*, originally introduced by Young and Yung [YY97, YY96b, YY96a]. A *kleptographic attack* "uses cryptography against cryptography" [YY97], by changing the behavior of a cryptographic system in a fashion undetectable to an honest user with black-box access to the cryptosystem, such that the use of the modified system leaks some secret information (*e.g.*, plaintexts or key material) to the attacker who performed the modification. An example of such an attack might be to modify the key generation algorithm of an encryption scheme such that an adversary in possession of a "back door" can derive the private key from the public key, yet an honest user finds the generated key pairs to be indistinguishable from correctly produced ones.

2.2 3SUM

The field of fine-grained complexity leverages algorithmic hardness assumptions to prove quantitative lower bounds for wide classes of problems. The standard assumptions in this field (Vassilevska Williams [VW15, VW18] gives excellent surveys of this topic) are hardness of CNF-SAT [IP01, IPZ01], 3SUM [GO95, Eri99], Orthogonal Vectors (OV) [Wil05], All Pairs Shortest Paths (APSP) [WW10], and Online Matrix-Vector Multiplication (OMV) [HKNS15].

Implications of the 3SUM Conjecture The 3SUM conjecture (Conjecture 1) has been helpful for understanding the precise hardness of many geometric problems [GO95, dBdGO97, BVKT98, ACH+98, Eri99, BHP01, AHI+01, SEO03, AEK05, EHPM06, CEHP07, AHP08, AAD+12]. Starting with the works of Vassilevska and Williams [VW09], and Pătraşcu [Păt10], the 3SUM conjecture has also been used for conditional lower bounds for many combinatorial [AVW14, GKLP16, KPP16] and string search [CHC09, BCC+13, AVWW14, ACLL14, AKL+16, KPP16] problems.

Algorithms for 3SUM The two standard algorithms for 3SUM on N integers from a range of size U run in time $O(N^2)$ and $O(N + U \log U)$ [CLRS09] (Exercise 30.1-7). The most efficient algorithm for this problem, due to Baran et al. [BDP08], runs in time $O(N^2(\log\log N)^2/\log^2 N)$. Kane et al. [KLM18] prove almost tight bounds on the linear decision tree complexity of 3SUM. Wang and Lincoln et al. [Wan14, LVWWW16] show that 3SUM algorithms can be implemented in space $\widetilde{O}(\sqrt{N})$. Also, Carmosino et al. [CGI+16] give a co-non-deterministic algorithm for 3SUM running in time $\widetilde{O}(N^{3/2})$, which shows that proving SETH-hardness of 3SUM is out of reach of the current techniques.

The "Real 3SUM" problem, where the input numbers are reals rather than integers, has also attracted a lot of attention. Starting with the breakthrough work of Grønlund and Pettie [GP14], the algorithms (in the Real-RAM model) for this version of 3SUM [Fre17, Cha18] achieved running time $O(N^2(\log\log N)^{O(1)}/\log^2 N)$, which essentially matches the running time of the algorithms for Integer 3SUM.

Data-structure versions of the conjecture While the standard conjectures about the hardness of CNF-SAT, 3SUM, OV and APSP concern *algorithms*, the OMV conjecture claims a *data structure* lower bound for the Matrix-Vector Multiplication problem. While algorithmic conjectures help to understand *time* complexity of the problems, it is also natural to consider data structure analogues of the fine-grained conjectures in order to understand *space* complexity of the corresponding problems. Recently Goldstein et al. [GKLP17, GLP17] proposed data structure variants of many classical hardness assumptions (including 3SUM and OV). Other data structure variants of the 3SUM problem have also been studied in [DV01, BW09, CL15, CCI+19]. In particular, Chan and Lewenstein [CL15] use techniques from additive combinatorics to give efficient data structures for solving 3SUM on *subsets* of the preprocessed sets.

3 Preliminaries

3.1 Notation

When an uppercase letter represents an integer, we use the convention that the associated lowercase letter represents its base-2 logarithm: $N = 2^n, S = 2^s$, etc. $[N]$ denotes the set $\{1, \dots, N\}$ that we identify with $\{0,1\}^n$. $x\|y$ denotes the concatenation of bit strings x and y. PPT stands for probabilistic polynomial time.

3.2 3SUM-Indexing

In this paper, we focus on the variant of 3SUM known as 3SUM-Indexing, formally defined in [GKLP17], which can be thought of as a preprocessing variant of 3SUM. Our definition is a generalization of [GKLP17], where we allow the input to be elements of an arbitrary abelian[3] group.

Definition 3. The problem 3SUM-Indexing(G, N), parametrized by an abelian group G and an integer N, is defined to be solved by a two-part algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ as follows.

- **Preprocessing phase.** $\mathcal{A}_1$ receives as input a tuple $A = (a_1, \dots, a_N)$ of N elements from G and outputs a data structure D_A of size[4] at most S. $\mathcal{A}_1$ is computationally unbounded.
- **Query phase.** Denote by $A + A$ the set of pairwise sums of elements from A: $A + A = \{a_i + a_j : 1 \le i < j \le N\}$. Given an arbitrary query $b \in A + A$, $\mathcal{A}_2$ makes at most T oracle queries to D_A and must output $(i, j) \in [N]^2$ with $i \ne j$ such that $a_i + a_j = b$.[5]

We say that $\mathcal{A}$ is an (S, T) algorithm for 3SUM-Indexing(G, N). Furthermore, we say that $\mathcal{A}$ is *non-adaptive* if the T queries made by $\mathcal{A}_2$ are non-adaptive (*i.e.*, the indices of the queried cells are only a function of b).

A few remarks about Definition 3 are in order.

[3] This is for convenience and because our applications only involve abelian groups; our results and techniques easily generalize to the non-abelian case.

[4] The model of computation in this paper is the word RAM model where we assume that the word length is $\Theta(\log N)$. Furthermore we assume that words are large enough to contain description of elements of G, *i.e.*, $|G| \le N^c$ for some $c > 0$. The size of a data structure is the number of words (or cells) it contains.

[5] Without loss of generality, we can assume that D_A contains a copy of S and in this case $\mathcal{A}_2$ could return a pair (a_i, a_j) at the cost of two additional queries.

Remark 1. An alternative definition would have the query b be an arbitrary element of G (instead of being restricted to $A+A$) and $\mathcal{A}_2$ return the special symbol $\perp$ when $b \in G \setminus (A+A)$. Again, an algorithm $\mathcal{A}$ for the problem as defined in Definition 3—with undefined behavior for $b \in G \setminus (A+A)$—can be turned into an algorithm for this seemingly more general problem at the cost of two extra queries: given output (i,j) on query b, return (i,j) if $a_i + a_j = b$ and return $\perp$ otherwise.

Remark 2. The requirement that $i \neq j$ for $\mathcal{A}_2$'s output is without loss of generality for integers, but prevents the occurrence of degenerate cases in some groups. For example, if G is such that all elements are of order 2 (*e.g.*, $(\mathbb{Z}/2\mathbb{Z})^{cn}$) then finding (i,j) such that $a_i + a_j = 0$ has the trivial solution $i = j$ for any $i \in [N]$.

Remark 3. In order to preprocess the elements of some group G, we assume an efficient way to enumerate its elements. More specificially, we assume a time- and space-efficient algorithm for evaluating an injective function $\mathsf{Index}\colon G \to [N^c]$ for a constant c. For simplicity, we also assume that the word length is at least $c \log N$ so that we can store $\mathsf{Index}(g)$ for every $g \in G$ in a memory cell. For example, for the standard 3SUM-Indexing problem over the integers from 0 to N^c, one can consider the group $G = (\mathbb{Z}/k\mathbb{Z}, +)$ for $k = 2N^c + 1$, and the trivial function $\mathsf{Index}(\overline{a_k}) = a$ for $0 \leq a < k$. For ease of exposition, we abuse notation and write g instead of $\mathsf{Index}(g)$ for an element of the group $g \in G$. For example, $g \mod p$ for an integer p will always mean $\mathsf{Index}(g) \mod p$.

The standard 3SUM-Indexing problem (formally introduced in [GKLP17]) corresponds to the case where $G = (\mathbb{Z}, +)$. In fact, it is usually assumed that the integers are upper-bounded by some polynomial in N, which is easily shown to be equivalent to the case where $G = (\mathbb{Z}/N^c\mathbb{Z}, +)$ for some $c > 0$, and is sometimes referred to as modular 3SUM when there is no preprocessing.

Another important special case is when $G = \big((\mathbb{Z}/2\mathbb{Z})^{cn}, +\big)$ for some $c > 0$. In this case, G can be thought of as the group of binary strings of length cn where the group operation is the bitwise XOR (exclusive or). This problem is usually referred to as 3XOR when there is no preprocessing, and we refer to its preprocessing variant as 3XOR-Indexing. In [JV16], the authors provide some evidence that the hardnesses of 3XOR and 3SUM are related and conjecture that Conjecture 1 generalizes to 3XOR. We similarly conjecture that in the presence of preprocessing, Conjecture 3 generalizes to 3XOR-Indexing.

Following Definition 3, the results and techniques in this paper hold for arbitrary abelian groups and thus provide a unified treatment of the 3SUM-Indexing and 3XOR-Indexing problems. It is an interesting open question for future research to better understand the influence of the group G on the hardness of the problem.

Open Question 1. *For which groups is 3SUM-Indexing significantly easier to solve, and for which groups does Conjecture 3 not hold?*

3.2.1 Average-case hardness

This paper moreover introduces a new average-case variant of 3SUM-Indexing (Definition 4 below) that, to the authors' knowledge, has not been stated in prior literature.[6] Definition 4 states an error parameter ε, as for the cryptographic applications it is useful to consider solvers for average-case 3SUM-Indexing that only output correct answers with probability $\varepsilon < 1$.

Definition 4. The *average-case* 3SUM-Indexing(G, N) problem, parametrized by an abelian group G and integer N, is defined to be solved by a two-part algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ as follows.

[6] We remark that for the classical version of 3SUM, the uniform random distribution of the inputs is believed to be the hardest (see, *e.g.*, [KPP16]).

- **Preprocessing phase.** Let A be a tuple of N elements from G drawn uniformly at random and with replacement. $\mathcal{A}_1(A)$ outputs a data structure D_A of size at most S. $\mathcal{A}_1$ has unbounded computational power.

- **Query phase.** Given a query b drawn uniformly at random in $A + A$, and given up to T oracle queries to D_A, $\mathcal{A}_2(b)$ outputs $(i, j) \in [N]^2$ with $i \neq j$ such that $a_i + a_j = b$.

We say that $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is an (S, T, ε) solver for 3SUM-Indexing if it answers the query correctly with probability ε over the randomness of $\mathcal{A}$, A, and the random query b. When $\varepsilon = 1$, we leave it implicit and write simply (S, T).

Remark 4. Note that in the query phase of Definition 4, the query b is chosen uniformly at random in $A+A$ and not in G. As observed in Remark 1, this is without loss of generality for $\varepsilon = 1$. When $\varepsilon < 1$, the meaningful way to measure the success probability of $\mathcal{A}$ is as in Definition 4 since $A+A$ could have negligible density in G and $\mathcal{A}$ could succeed with overwhelming probability by always outputting $\perp$.

4 Upper bound

We will use the following data structure first suggested by Hellman [Hel80] and then rigorously studied by Fiat and Naor [FN00].

Theorem 5 ([FN00])**.** *For any function $F\colon D \to D$, and for any choice of values S and T such that $S^3 T \geq |D|^3$, there is a deterministic data structure with space $\widetilde{O}(S)$ which allows inverting F at every point making $\widetilde{O}(T)$ queries to the memory cells and evaluations of F.* [7]

The idea of our upper bound is the following. Since we are only interested in the pairwise sums of the N input elements $a_1, \ldots, a_N \in G$, we can hash down their sums to a set of size $O(N^2)$. Now we define the function $f(i, j) = a_i + a_j$ for $i, j \in [N]$, and note that its domain and range are both of size $O(N^2)$. We apply the generic inversion algorithm of Fiat and Naor to f with $|D| = O(N^2)$, and obtain a data structure for 3SUM-Indexing.

First, in Lemma 6 we give an efficient data structure for the "modular" version of 3SUM-Indexing(G, N) where for an integer $p = \widetilde{O}(N^2)$ and N inputs $a_1, \ldots, a_N \in G$, each query $b \in G$ asks to find $i, j \in [N]$ such that $a_i + a_j \equiv b \mod p$.[8] Then, in Theorem 7 we reduce the general case of 3SUM-Indexing(G, N) to the modular one.

Lemma 6. *For every $0 \leq \delta \leq 1$ and every integer $p = \widetilde{O}(N^2)$, there is an* adaptive *data structure which uses space $S = \widetilde{O}(N^{2-\delta})$ and query time $T = \widetilde{O}(N^{3\delta})$ and solves modular 3SUM-Indexing(G, N): for input $a_1, \ldots, a_N \in G$ and a query $b \in G$, it outputs a_i, a_j such that $a_i + a_j \equiv b \mod p$, if such a_i and a_j exist.*

Proof. Let the N input elements be $a_1, \ldots, a_N \in G$. The data structure will store all a_i (this takes only N memory cells) along with the information needed to efficiently invert the function $f\colon [N] \times [N] \to G$ defined below. For $i, j \in G$, let $f(i, j) = a_i + a_j \mod p$. Note that:

1. f is easy to compute. Indeed, given the input, one can compute f by looking at only two input elements.

[7] While the result in Theorem 1.1 in [FN00] is stated for a randomized preprocessing procedure, we remark that a less efficient *deterministic* procedure which brute forces the probability space can be used instead.

[8] Recall from Remark 3 that this notation actually means $\mathsf{Index}(a_i + a_j) \equiv \mathsf{Index}(b) \mod p$.

2. The domain of f is of size N^2, and the range of f is of size $p = \widetilde{O}(N^2)$.
3. Inverting f at a point $b \in G$ allows one to check whether there exists a_i and a_j such that $a_i + a_j \equiv b \mod p$, which essentially solves the modular $\mathsf{3SUM\text{-}Indexing}(G, N)$ problem.

Now we use the data structure from Theorem 5 with $|D| = \widetilde{O}(N^2)$ to invert f. This gives us a data structure with space $\widetilde{O}(S+N) = \widetilde{O}(S)$ and query time $\widetilde{O}(T)$ for every $S^3T \geq |D|^3 = \widetilde{O}(N^6)$, which finishes the proof. □

It remains to show that the input of 3SUM-Indexing can always be hashed to a set of integers $[D]$ for some $D = \widetilde{O}(N^2)$. While many standard hashing functions will work here, we remark that it is important for our applciation that the hash function of choice has a time- and space-efficient implementation (for example, the data structure in [FN00] requires non-trivial implementations of hash functions). Below we present a simple hashing procedure which suffices for 3SUM-Indexing, but a more general reduction can be found in Lemma 17 in [CGK18].

Theorem 7. *For every $0 \leq \delta \leq 1$, there is an* adaptive *data structure for 3SUM-Indexing(G, N) with space $S = \widetilde{O}(N^{2-\delta})$ and query time $T = \widetilde{O}(N^{3\delta})$.*

In particular, by taking $\delta = 0.1$, we get a data structure which solves 3SUM-Indexing in space $S = \widetilde{O}(N^{1.9})$ and query time $T = \widetilde{O}(N^{0.3})$, and, thus, refutes Conjecture 4.

Proof. Let the N inputs be $a_1, \ldots, a_N \in G$. Let $Z \subseteq [N^c], |Z| < N^2$ be the set of pairwise sums of the inputs:

$$Z \coloneqq \{a_i + a_j \colon 1 \leq i \leq j \leq N\}\,.$$

Let $I = \{N^2, \ldots, 5cN^2 \log N\}$ be an interval of integers. By the prime number theorem, for large enough N, I contains at least $4cN^2$ primes. Let us pick $m = \log N$ random primes $p_1, \ldots, p_m$ from I. For two distinct numbers $z_1, z_2 \in Z$, we say that they have a collision modulo p if p divides $z_1 - z_2$.

Let $g \in G$ be a positive query of $\mathsf{3SUM\text{-}Indexing}(G, N)$, that is, $b = \mathsf{Index}(g) \in Z$. First, we show that with high probability (over the choices of m random primes) there exists an $i \in [m]$ such that for every $z \in Z, z \neq b$, b and z do not collide modulo p_i. Indeed, for every $z \in Z, z \neq b$, we have that $(z - b)$ has at most $c/2$ prime factors from I. Since $|Z| < N^2$, at most $cN^2/2$ primes from I divide $(z - b)$ for *any* $z \in Z$. Therefore, a random prime from I gives a collision for b and some z with probability at most $1/8$. Now we have that for every $b \in Z$, the probability that there exists an $i \in [m]$ such that b does not collide with any $z \in Z, z \neq b$ modulo p_i, is at least $1 - (1/8)^m = 1 - N^{-3}$. Therefore, with high probability, a random set of m primes works for all $b \in Z$: for every b there exists an $i \in [m]$ such that b does not collide with any $z \in Z, z \neq b$ modulo p_i. Since such a set of m primes exists, the preprocessing stage of the data structure can find it deterministically.

Now we construct $m = \log N$ modular $\mathsf{3SUM\text{-}Indexing}(G, N)$ data structures (one for each p_i), and separately solve the problem for each of the m primes. This results in a data structure as guaranteed by Lemma 6 with a $\log N$ overhead in space and time. The data structure also stores the inputs $a_1, \ldots, a_N$. Once it sees a solution modulo p_i, it checks whether it corresponds to a solution to the original problem. Now correctness follows from two observations. Since the data structure checks whether a solution modulo p_i gives a solution to the original problem, the data structure never reports false positives. Second, the above observation that for every $b \in Z$ there is a prime p_i such that b does not collide with other $z \in Z$, a solution modulo p_i will correspond to a solution of the original problem (thus, no false negatives can be reported either). □

Remark 5. A few extensions of Theorem 7 are in order.

1. The result of Fiat and Naor [FN00] also gives an efficient *randomized* data structure. Namely, there is a randomized data structure with preprocessing running time $\widetilde{O}(|D|)$, which allows inverting F at *every* point with probability at least $1 - 1/|D|$ over the randomness of the preprocessing stage. Thus, the preprocessing phase of the randomized version of Theorem 5 runs in quasilinear time $\widetilde{O}(|D|) = \widetilde{O}(N^2)$ (since sampling $m = \log N$ random primes from a given interval can also be done in randomized time $\widetilde{O}(1)$). This, in particular, implies that the preprocessing time of the presented data structure for 3SUM-Indexing is optimal under the 3-SUM Conjecture (Conjecture 1). Indeed, if the preprocessing time was improved to $N^{2-\varepsilon}$, then one could solve 3-SUM by querying the N input numbers in (randomized or expected) time $N^{2-\varepsilon}$.

2. We remark that for polynomially small $\varepsilon = 1/|D|^\alpha$ (for constant α), the trade-off between S and T can be further improved for the ε-approximate solution of 3SUM-Indexing, using approximate function inversion by De et al. [DTT10].

We showed how to refute the strong 3SUM-Indexing conjecture of [GKLP17] using techniques from space-time tradeoffs for function inversion [Hel80, FN00], specifically the general function inversion algorithm of Fiat and Naor [FN00]. A natural open question is whether a more specific function inversion algorithm could be designed:

Open Question 2. *Can the space-time trade-off achived in Theorem 7 be improved by exploiting the specific structure of the 3SUM-Indexing problem?*

5 Lower bound

We now present our lower bound: we prove a space-time trade-off of $S = \widetilde{\Omega}(N^{1+1/T})$ for any non-adaptive (S,T) algorithm. While it is weaker than Conjecture 3, any improvement on this result would break a long-standing barrier in static data structure lower bounds: no bounds better than $T \geq \Omega(\frac{\log N}{\log(S/N)})$ are known, even for non-adaptive cell-probe and linear models [Sie04, Pǎt11, PTW10, Lar12, DGW19].

Our proof relies on a compressibility argument similar to [GT00, DTT10] also known as cell-sampling in the data structure literature [PTW10]. Roughly speaking, we show that given an (S,T) algorithm $(\mathcal{A}_1, \mathcal{A}_2)$, we can recover a subset of the input A by storing a randomly sampled subset V of the preprocessed data structure D_A and simulating $\mathcal{A}_2$ on all possible queries: the simulation succeeds whenever the queries made by $\mathcal{A}_2$ fall inside V. Thus, by storing V along with the remaining part of the input, we obtain an encoding of the entire input. This implies that the length of the encoding must be at least the entropy of a randomly chosen input.

Theorem 8. *Let $N \geq 2$ be an integer and G be an abelian group with $|G| \geq N^2$, then any non-adaptive (S,T) algorithm for 3SUM-Indexing(G, N) satisfies: $S = \widetilde{\Omega}(N^{1+1/T})$.*

Proof. Consider an (S,T) algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for 3SUM-Indexing(G, N). We want to use $\mathcal{A}$ to design encoding and decoding procedures for inputs of 3SUM-Indexing(G, N). For this, we will first sample a subset V of the data structure cells which allows us to answer many queries. Using this set, we will argue that we can recover a constant fraction of the input, which will lead to a succinct encoding of the input.

Sampling a subset V of cells. For a query $b \in G$, $Q(b) \subseteq [S]$ denotes the set of probes made by $\mathcal{A}_2$ on input b (with $|Q(b)| \leq T$, since $\mathcal{A}_2$ makes at most T probes to the data structure). Given a subset $V \subseteq [S]$ of cells, we denote by G_V the set of queries in G which can be answered by $\mathcal{A}_2$ by only making probes within V: $G_V = \{b \in G : Q(b) \subseteq V\}$. Observe that for a uniformly random set V of size v:

$$\mathbb{E}\big[|G_V|\big] = \sum_{b \in G} \Pr[Q(b) \subseteq V] \geq |G| \frac{\binom{S-T}{v-T}}{\binom{S}{v}} = |G| \prod_{i=0}^{T-1} \frac{v-i}{S-i} \geq |G| \left(\frac{v-T}{S-T}\right)^T ,$$

where the last inequality uses that $a/b \geq (a-1)/(b-1)$ for $a \leq b$. Hence, there exists a subset V of size v, such that:

$$|G_V| \geq |G| \left(\frac{v-T}{S-T}\right)^T ,$$

and we will henceforth consider such a set V. The size v of V will be set later so that $|G_V| \geq |G|/N$.

Using V to recover the input. Consider some input $A = (a_1, \ldots, a_N)$ for $\mathsf{3SUM\text{-}Indexing}(G, N)$. We say that $i \in [N]$ is *good* if a_i is output by $\mathcal{A}_2$ given some query in G_V. Since queries in G_V can be answered by only storing the subset of cells of the data structure indexed by V, our decoding procedure will retrieve from these cells all the good elements from A. Note that $i \in [N]$ is good if:

$$\exists j \in [N] \setminus \{i\},\ (a_i + a_j \in G_V) \wedge \big(\forall k \in [N] \setminus \{i\}, \forall \ell \in [N] \setminus \{i, k\},\ a_k + a_\ell \neq a_i + a_j\big) . \tag{1}$$

Indeed, observe that:

1. The first part of the conjunction guarantees that there exists $b \in G_V$ which can be decomposed as $b = a_i + a_j$ for $j \neq i$.
2. The second part of the conjunction guarantees that the decomposition $b = a_i + a_j$ is unique, *i.e.* that there is no other pair of elements of A which adds up to b.

By correctness of $\mathcal{A}$, $\mathcal{A}_2$ outputs a decomposition of its input as a sum of two elements in A if one exists. For j as in (1), the decomposition $b = a_i + a_j$ is unique and hence $\mathcal{A}_2(a_i + a_j) = (a_i, a_j)$.

We denote by $N_V \subseteq [N]$ the set of good indices, and compute its expected size when A is chosen at random according to the distribution in Definition 4, *i.e.* for each $i \in [N]$, a_i is chosen independently and uniformly in G.

$$\mathbb{E}\big[|N_V|\big] \geq \sum_{i=1}^{N} \Pr[\exists j \neq i, a_i + a_j \in G_V] \Pr[\forall k, \ell \neq i, a_k + a_\ell \neq a_i + a_j \mid a_i + a_j \in G_V] \tag{2}$$

Note that:

$$\Pr[\exists j \neq i, a_i + a_j \in G_V] = 1 - \Pr[\forall j \neq i, a_i + a_j \notin G_V] = 1 - \left(1 - \frac{|G_V|}{|G|}\right)^{N-1} ,$$

where the second equality is because for $j \neq i$, a_j needs to be distinct from the $|G_V|$ elements $-a_i + g$ for $g \in G_V$. Furthermore:

$$\begin{aligned}
\Pr[\forall k, \ell \neq i, a_k + a_\ell \neq a_i + a_j \mid a_i + a_j \in G_V] &= 1 - \Pr[\exists k, \ell \neq i, a_k + a_\ell = a_i + a_j \mid a_i + a_j \in G_V] \\
&\geq 1 - \sum_{k, \ell \neq i} \Pr[a_k + a_\ell = a_i + a_j \mid a_i + a_j \in G_V] \\
&\geq 1 - \frac{(N-1)(N-2)}{2} \frac{1}{|G|} \geq \frac{1}{2} ,
\end{aligned}$$

where the first inequality uses the union bound and the last inequality uses that $|G| \geq N^2$. Using the previous two derivations in (2), we get:

$$\mathbb{E}\big[|N_V|\big] \geq \frac{N}{2}\left(1 - \left(1 - \frac{|G_V|}{|G|}\right)^{N-1}\right) \geq \frac{N}{4}\,, \tag{3}$$

where the last inequality uses that $|G_V| \geq |G|/N$ and that $(1-1/N)^{N-1} \leq 1/2$ for $N \geq 2$.

Encoding and decoding. It follows from (3) and a simple averaging argument that with probability at least $1/16$ over the random choice of A, N_V is of size at least $N/5$. We will henceforth focus on providing encoding and decoding procedures for such inputs A. Specifically, consider the following pair of encoding/decoding algorithms for A:

- $\mathsf{Enc}(A)$: given input $A = (a_1, \ldots, a_N)$.
  1. use $\mathcal{A}_2$ to compute the set $N_V \subseteq [N]$ of good indices.
  2. store $\big(\mathcal{A}_1(A)_j\big)_{j \in V}$ and $(a_i)_{i \notin N_V}$.
- $\mathsf{Dec}\big(\mathsf{Enc}(A)\big)$: for each $b \in G$, simulate $\mathcal{A}_2$ on input b:
  1. If $Q(b) \subseteq V$, use $\big(\mathcal{A}_1(A)_i\big)_{i \in V}$ (which was stored in $\mathsf{Enc}(A)$) to simulate $\mathcal{A}_2$ and get $\mathcal{A}_2(b)$. By definition of N_V, when b ranges over the queries such that $Q(b) \subseteq V$, this step recovers $(a_i)_{i \in N_V}$.
  2. Then recover $(a_i)_{i \notin N_v}$ directly from $\mathsf{Enc}(A)$.

Note that the bit length of the encoding is:

$$|\mathsf{Enc}(A)| \leq v \cdot w + (N - |N_V|) \log |G| \leq v \cdot w + \frac{4N}{5} \log |G|$$

where w is the word length and where the second inequality holds because we restrict ourselves to inputs A such that $|N_V| \geq N/5$. By a standard incompressibility argument (see for example Fact 8.1 in [DTT10]), since our encoding and decoding succeeds with probability at least $1/16$ over the random choice of A, we need to be able to encode at least $|G|^N/16$ distinct values, hence:

$$v \cdot w + \frac{4N}{5} \log |G| \geq N \log |G| + O(1) \tag{4}$$

Finally, as discussed before, we set v such that $|G_V|/|G| \geq 1/N$. For this, by the computation performed at the beginning of this proof, it is sufficient to have:

$$\left(\frac{v-T}{S-T}\right)^T \geq \frac{1}{N}\,.$$

Hence, we set $v = T + (S-T)/N^{1/T}$ and since $T \leq N \leq S$ (otherwise the result is trivial), (4) implies:

$$S = \widetilde{\Omega}(N^{1+1/T}) \qquad \square$$

Remark 6. $\mathsf{3SUM\text{-}Indexing}(\mathbb{Z}/N^c\mathbb{Z}, N)$ reduces to $\mathsf{3SUM\text{-}Indexing}$ over the integers, so our lower bound extends to $\mathsf{3SUM\text{-}Indexing}(\mathbb{Z}, N)$, too. Specifically, the reduction works as follows: we choose $\{\bar{0}, \ldots, \overline{N^c - 1}\}$ as the set of representatives of $\mathbb{Z}/N^c\mathbb{Z}$. Given some input $A \subseteq \mathbb{Z}/N^c\mathbb{Z}$ for $\mathsf{3SUM\text{-}Indexing}(\mathbb{Z}/N^c\mathbb{Z}, N)$, we treat it as a list of integers and build a data structure using our algorithm for $\mathsf{3SUM\text{-}Indexing}(\mathbb{Z}, N)$. Now, given a query $\bar{b} \in \mathbb{Z}/N^c\mathbb{Z}$, we again treat it as an integer and query the data structure at b and $b + N^c$. The correctness of the reduction follows from the observation that $\bar{b} = \bar{a}_i + \bar{a}_j$ if and only if $a_i + a_j = b$ or $a_i + a_j = b + N^c$.

As we already mentioned, no lower bound better than $T \geq \Omega(\frac{\log N}{\log(S/N)})$ is known even for non-adaptive cell-probe and linear models, so Theorem 8 matches the best known lower bounds for static data structures. An ambitious goal for future research would naturally be to prove Conjecture 3. A first step in this direction would be to extend Theorem 8 to adaptive strategies that potentially err with some probability.

Open Question 3. *Must any (possibly adaptive) (S, T, ε) algorithm for 3SUM-Indexing(G, N) require $S = \tilde{\Omega}(\varepsilon N^{1+1/T})$?*

6 Application to cryptography

6.1 Background on random oracles and preprocessing

A line of work initiated by [IR89] studies the hardness of a random oracle as a one way function. In [IR89] it was shown that a random oracle is an exponentially hard one-way function against uniform adversaries. The case of non-uniform adversaries was later studied in [Imp96, Zim98]. Specifically we have the following result.

Proposition 9 ([Zim98])**.** *With probability at least $1 - \frac{1}{N}$ over the choice of a random oracle $R : \{0,1\}^n \to \{0,1\}^n$, for all oracle circuits C of size at most T:*

$$\Pr_{x \leftarrow \{0,1\}^n} \left[C^R\big(R(x)\big) \in R^{-1}\big(R(x)\big) \right] \in \widetilde{O}\left(\frac{T^2}{N}\right) .$$

In Proposition 9, the choice of the circuit occurs *after* the random draw of the oracle: in other words, the description of the circuit can be seen as a non-uniform advice which depends on the random oracle. Proposition 10 is a slight generalization where the adversary is a uniform Turing machine independent of the random oracle, with oracle access to an advice of length at most S depending on the random oracle. While the two formulations are equivalent in the regime $S \leq T$, one advantage of this reformulation is that S can be larger than the running time T of the adversary.

Proposition 10 (Implicit in [DTT10])**.** *Let $\mathcal{A}$ be a uniform oracle Turing machine whose number of oracle queries is $T : \{0,1\}^n \to \mathbb{N}$. For all $n \in \mathbb{N}$ and $S \in \mathbb{N}$, with probability at least $1 - \frac{1}{N}$ over the choice of a random oracle $R : \{0,1\}^n \to \{0,1\}^n$:*

$$\forall\, P \in \{0,1\}^S, \Pr_{x \leftarrow \{0,1\}^n} \left[\mathcal{A}^{R,P}\big(R(x)\big) \in R^{-1}\big(R(x)\big) \right] \in O\left(\frac{T(S+n)}{N}\right) .$$

In Proposition 10, the advice P can be thought of as the result of a preprocessing phase involving the random oracle. Also, no assumption is made on the computational power of the preprocessing adversary but it is simply assumed that the length of the advice is bounded.

Remark 7. Propositions 9 and 10 assume a deterministic adversary. For the regime of $S > T$ (which is the focus of this work), this assumption is without loss of generality since a standard averaging argument shows that for a randomized adversary, there exist a choice of "good" randomness for which the adversary achieves at least its expected success probability. This choice of randomness can be hard-coded in the non-uniform advice, yielding a deterministic adversary.

Note, however, that Proposition 10 provides no guarantee when $S \geq N$. In fact, in this case, defining P to be any inverse mapping R^{-1} of R allows an adversary to invert R with probability one by making a single query to P. So, R itself can no longer be used as a one-way function when $S \geq N$ — but one can still hope to use R to define a new function f^R that is one-way against an adversary with advice of size $S \geq N$. This idea motivates the following definition.

Definition 11. Let $R : \{0,1\}^n \to \{0,1\}^n$ be a random oracle. A *one-way function in the random oracle model with S preprocessing* is an efficiently computable oracle function $f^R : \{0,1\}^n \to \{0,1\}^{n'}$ such that for any two-part adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ satisfying $|\mathcal{A}_1(\cdot)| \le S$ and where $\mathcal{A}_2$ is PPT, the following probability is negligible in n:[9]

$$\Pr_{R, x \leftarrow \{0,1\}^n}\left[f^R\left(\mathcal{A}_2^{R, \mathcal{A}_1(R)}\left(f^R(x)\right)\right) = f^R(x) \right] .$$

The adversary model in Definition 11 is almost identical to the 1-BRO model of [BFM18], differing only in having a restriction on the output size of $\mathcal{A}_1$. As was noted in [BFM18], without this restriction (and in fact, as soon as $S \ge 2^{n'}$ by the same argument as above), no function f^R can achieve the property given in Definition 11. [BFM18] bypasses this impossibility by considering the restricted case of two independent oracles with two independent preprocessed advices (of unrestricted sizes). Our work bypasses it in a different and incomparable way, by considering the case of a single random oracle with bounded advice.

6.2 Candidate construction

Our main candidate construction of a OWF (Construction 13) relies on the hardness of average-case 3SUM-Indexing. First, we define what hardness means, then give the constructions and proofs.

Definition 12. Average-case 3SUM-Indexing is $(G, N, S, T, \varepsilon)$*-hard* if the success probability[10] of any (S, T) algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in answering average-case 3SUM-Indexing(G, N) queries is at most ε.

Construction 13. Let N be an integer and G be an abelian group with $|G| \ge N$ and let $R : [N] \to G$ be a random oracle.

Our candidate OWF construction has two components:

- the function $f^R : [N] \times [N] \to G$ defined as follows, where $+$ denotes G's operation:

$$f^R(i, j) = R(i) + R(j),\ (i, j) \in [N]^2 .$$

- the input distribution (I, J) where I is uniformly random in $[N]$ and J is uniformly random in $[N] \setminus \{I\}$.

Remark 8 (Approximate sampling). For convenience, our candidate OWF is defined with respect to an input distribution which is not uniform over the domain of f^R. While this may seem restrictive, as long as the input distribution is efficiently samplable, a standard construction shows how to transform a one way function of this type into a one-way function which operates on uniformly random bit strings (see [**?**, Section 2.4.2]).

In our case, since $N(N-1)$ is not a power of two (for $N > 2$), the input distribution (I, J) in Construction 13 cannot be sampled exactly in time polynomial in $\log N$. However, using rejection sampling, it is easy to construct a sampler taking as input $\lceil \log N \rceil^2$ random bits and whose output distribution is $1/N$ close to (I, J) in statistical distance. It is easy to propagate this (negligibly) small sampling error without affecting the conclusion of Theorem 14.

This is similar to what happens when considering one way functions based on the hardness of factoring which require sampling integers in a range whose length is not necessarily a power of two.

[9] A negligible function is one that is in $o(n^{-c})$ for all constants c.

[10] Over the randomness of $\mathcal{A}$, A, and the average-case 3SUM-Indexing query. (Recall: A is 3SUM-Indexing's input.)

Remark 9. Similarly, the random oracle R used in the construction is not a random oracle in the traditional sense since its domain and co-domain are not bit strings. If $|G|$ and N are powers of two, then R can be implemented exactly by a standard random oracle $\{0,1\}^{\log N} \to \{0,1\}^{\log |G|}$. If not, using a random oracle $\{0,1\}^{\mathsf{poly}(\lceil \log |G| \rceil)} \to \{0,1\}^{\mathsf{poly}(\lceil \log |G| \rceil)}$, and rejection sampling, it is possible to implement an oracle R' which is $1/N$ close to R in statistical distance. We can similarly propagate this $1/N$ sampling error without affecting the conclusion of Theorem 14.

Theorem 14. *Consider a sequence of abelian groups $(G_N)_{N\geq 1}$ such that $|G_N| \geq N^{2+c}$ for some $c > 0$ and all $N \geq 1$, and a function $S : \mathbb{N} \to \mathbb{R}$. Assume that for all polynomial T there exists a negligible function ε such that average-case* 3SUM-Indexing *is $(G_N, N, S(n), T(n), \varepsilon(n))$ hard for all $N \geq 1$ (recall that $n = \log N$). Then the function f defined in Construction 13 is a one-way function in the random oracle model with S preprocessing.*

The function f^R in Construction 13 is designed precisely so that inverting f^R is equivalent to solving 3SUM-Indexing for the input $A = (R(1), \dots, R(N))$. However, the one-way function inversion game and average-case 3SUM-Indexing are not exactly identical. Indeed, in the one-way function inversion game, the input to the adversary is the random variable $f^R(I, J)$ which is distributed as $A_I + A_J$ for a uniformly random set of indices $\{I, J\}$. In contrast, in average-case 3SUM-Indexing, the input to the adversary is uniform over $A + A$. These two input distributions are not identical if there are collisions: pairs of sets $\{i, j\}$, $\{k, l\}$ such that $A_i + A_j = A_k + A_l$. The following two lemmas show that whenever $|G| \geq N^{2+c}$ for some $c > 0$, there are sufficiently few collisions that the two input distributions are negligibly close in statistical distance, which is sufficient to prove Theorem 14.

Lemma 15. *Let $N \geq 2$ be an integer and let G be an abelian group with $|G| \geq N^{2+c}$ for some $c > 0$. Let $A = (A_1, \dots, A_N)$ be a tuple of N elements drawn with replacement from G. Define the following two random variables:*

- *$X_1 = A_I + A_J$ where $\{I, J\} \subseteq [N]$ is a uniformly random set of size 2.*
- *X_2: uniformly random in $A + A = \{A_i + A_j : 1 \leq i < j \leq N\}$.*

Then the statistical distance $\|(A, X_1) - (A, X_2)\|_{\mathsf{s}} = O\big(1/\sqrt{N^c}\big)$.

Proof. First, by conditioning on the realization of A:

$$\|(A, X_1) - (A, X_2)\|_{\mathsf{s}} = \sum_{a \in G^N} \Pr[A = a] \|X_{1|a} - X_{2|a}\|_{\mathsf{s}}\,, \tag{5}$$

where $X_{i|a}$ denotes the distribution of X_i conditioned on the event $A = a$ for $i \in \{1, 2\}$.

We now focus on a single term, corresponding to the realization $A = a$ and define the set of pairwise sums $D = \{a_i + a_j : 1 \leq i < j \leq N\}$, and for $g \in G$, $c_g = \big|\big\{\{i, j\} \subseteq [N]^2 : a_i + a_j = g\big\}\big|$ is the number of pairs whose sum is g. Then we have:

$$\|X_{1|a} - X_{2|a}\|_{\mathsf{s}} = \frac{1}{2} \sum_{g \in D} \left| \frac{1}{|D|} - \frac{c_g}{\binom{N}{2}} \right| .$$

Observe that $c_g \geq 1$ whenever $g \in D$. We now assume that $|D| \geq \frac{1}{2}\binom{N}{2}$ (we will later only use the following derivation under this assumption). Splitting the sum on $c_g > 1$:

$$\|X_{1|a} - X_{2|a}\|_{\mathsf{s}} = \frac{1}{2} \sum_{g\,:\,c_g = 1} \left(\frac{1}{|D|} - \frac{1}{\binom{N}{2}} \right) + \frac{1}{2} \sum_{g\,:\,c_g > 1} \left(\frac{c_g}{\binom{N}{2}} - \frac{1}{|D|} \right) ,$$

where we used the trivial upper bound $|D| \leq \binom{N}{2}$ and the assumption that $|D| \geq \frac{1}{2}\binom{N}{2}$ to determine the sign of the quantity inside the absolute value. We then write:

$$\begin{aligned}
\|X_{1|a} - X_{2|a}\|_{\mathsf{s}} &= \frac{1}{2} \sum_{g\,:\,c_g=1} \left(\frac{1}{|D|} - \frac{1}{\binom{N}{2}} \right) + \frac{1}{2} \sum_{g\,:\,c_g>1} \left(\frac{c_g - 1}{\binom{N}{2}} + \frac{1}{\binom{N}{2}} - \frac{1}{|D|} \right) \\
&\leq \frac{1}{2} \sum_{g\,:\,c_g\geq 1} \left(\frac{1}{|D|} - \frac{1}{\binom{N}{2}} \right) + \frac{1}{2} \sum_{g\,:\,c_g>1} \frac{c_g - 1}{\binom{N}{2}} \\
&= \frac{1}{2} \sum_{g\,:\,c_g\geq 1} \left(\frac{1}{|D|} - \frac{1}{\binom{N}{2}} \right) + \frac{1}{2} \sum_{g\,:\,c_g\geq 1} \frac{c_g - 1}{\binom{N}{2}} = \left(1 - \frac{|D|}{\binom{N}{2}} \right),
\end{aligned}$$

where the inequality uses again that $|D| \leq \binom{N}{2}$, and the last equality uses that $\sum_{g:c_g\geq 1} c_g = \binom{N}{2}$ equals the number of pairwise sums.pairwise sums is $D = \{g \,:\, c_g \geq 1\}$.

We now consider some $\delta \leq 1/2$ which will be set at the end of the proof and split the sum in (5) on $|D| \leq (1-\delta)\binom{N}{2}$:

$$\begin{aligned}
\|(A, X_1) - (A, X_2)\|_{\mathsf{s}} &\leq \Pr\left[|D| \leq \binom{N}{2}(1-\delta) \right] + \delta \cdot \Pr\left[|D| > \binom{N}{2}(1-\delta) \right] \\
&\leq \Pr\left[|D| \leq \binom{N}{2}(1-\delta) \right] + \delta\,,
\end{aligned}$$

where we used the trivial upper bound $\|X_{1|a} - X_{2|a}\|_{\mathsf{s}} \leq 1$ when $|D| \leq (1-\delta)\binom{N}{2}$ and the upper bound $\|X_{1|a} - X_{2|a}\|_{\mathsf{s}} < \delta$ when $|D| > (1-\delta)\binom{N}{2}$ by the previous derivation.

We now use Markov's inequality and Lemma 16 to upper bound the first summand:

$$\begin{aligned}
\|(A, X_1) - (A, X_2)\|_{\mathsf{s}} &\leq \frac{1}{\delta\binom{N}{2}} \left(\binom{N}{2} - \mathbb{E}\big[|D|\big] \right) + \delta \\
&\leq \frac{1}{\delta |G|} \binom{N}{2} + \delta \leq \frac{1}{2\delta N^c} + \delta\,.
\end{aligned}$$

where the last inequality uses that $|G| \geq N^{2+c}$ by assumption. Finally, we set $\delta = 1/\sqrt{N^c}$ to get the desired conclusion. □

Lemma 16. *Let G be an abelian group and let $A = (a_1, \ldots, a_N)$ be a tuple of N elements drawn with replacement from G. Define $D = \big\{a_i + a_j \,:\, 1 \leq i < j \leq N\big\}$ to be the set of pairwise sums of coordinates of A, then:*

$$\binom{N}{2} - \mathbb{E}\big[|D|\big] \leq \frac{1}{|G|} \binom{N}{2}^2.$$

Proof. For each set of two indices $\{i, j\} \subseteq [N]$, we define the random variable $X_{\{i,j\}}$ to be the indicator that the sum $a_i + a_j$ collides with $a_k + a_l$ for some $\{k, l\} \neq \{i, j\}$:

$$X_{\{i,j\}} = \mathbf{1}\big\{ \exists \{k, l\} \neq \{i, j\} \,:\, a_k + a_l = a_i + a_j \big\}\,.$$

Then, using a union bound and since the probability of a collision is $1/|G|$:

$$\mathbb{E}\big[X_{\{i,j\}}\big] \leq \sum_{\{k,l\}\neq\{i,j\}} \Pr\left[a_i + a_j = a_k + a_l\right] \leq \frac{\binom{N}{2}}{|G|}\,.$$

On the other hand, there are at least as many elements in D as sets of indices $\{i,j\} \subseteq [N]$ which do not collide:

$$D \geq \sum_{\{i,j\}\subseteq[N]} (1 - X_{\{i,j\}}) = \binom{N}{2} - \sum_{\{i,j\}\subseteq[N]} X_{\{i,j\}} \,.$$

Combining the previous two inequalities concludes the proof. □

We are now ready to prove Theorem 14.

Proof (Theorem 14). Throughout the proof, we fix N and write G, S, T to denote $G_N, S(n), T(n)$ respectively, leaving the parameter n implicit. Suppose, for contradiction, that f is not a one-way function in the random oracle model with S preprocessing. Then there exists $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ such that $|\mathcal{A}_1(\cdot)| \leq S$ and $\mathcal{A}_2$ is PPT, which inverts f with probability at least δ for some non-negligible δ:

$$\Pr_{R,I,J}\left[f^R\left(\mathcal{A}_2^{R,\mathcal{A}_1(R)}\left(f^R(I,J)\right)\right) = f^R(I,J)\right] \geq \delta \,. \tag{6}$$

where $R : [N] \to G$ is a random oracle and (I,J) is the distribution of inputs to f^R as defined in Construction 13. Then we use $\mathcal{A}$ to build an (S,T) solver $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ for $\mathsf{3SUM\text{-}Indexing}(G,N)$ as follows. Given input $A = (A_1, \ldots, A_N)$ for $\mathsf{3SUM\text{-}Indexing}(G,N)$, $\mathcal{A}'_1$ defines random oracle $R : [N] \to G$ such that $R(i) = A_i$ for $i \in [N]$ and outputs $\mathcal{A}_1(R)$—this amounts to interpreting the tuple as a function mapping indices to coordinates. $\mathcal{A}'_2$ is identical to $\mathcal{A}_2$. By construction, whenever $\mathcal{A}_2$ successfully inverts f (*i.e.*, outputs (i,j) such that $f^R(i,j) = b$ for input b), then the output of $\mathcal{A}'_2$ satisfies $A_i + A_j = b$.

It follows from (6) that $\mathcal{A}'$ as described thus far solves average-case $\mathsf{3SUM\text{-}Indexing}(G,N)$ with success probability δ when given as input a query distributed as $f^R(I,J)$. By construction, the distribution of $f^R(I,J)$ is identical to the distribution of $A_I + A_J$ for uniformly random set $\{I,J\} \subseteq [N]$, let X_1 denote this distribution. However, average-case $\mathsf{3SUM\text{-}Indexing}(G,N)$ is defined with respect to a distribution of queries which is uniform over $A + A$, let us denote this distribution by X_2. By Lemma 15, we have that $\|(A, X_1) - (A, X_2)\|_{\mathsf{s}} = O(1/\sqrt{N^c})$, hence $\mathcal{A}_2$ solves $\mathsf{3SUM\text{-}Indexing}(G,N)$ for the correct query distribution X_2 with probability at least $\delta - O(1/\sqrt{N^c})$ which is non-negligible since δ is non-negligible. Denoting by T the running time of $\mathcal{A}_2$, we just proved that $\mathcal{A}'$ is an $(S, T, \delta - O(1/\sqrt{N^c}))$ adversary for average-case $\mathsf{3SUM\text{-}Indexing}(G,N)$, which is a contradiction. □

We conjecture that $\mathsf{3SUM\text{-}Indexing}$ is $(G, N, S, T, \varepsilon)$-hard with $\varepsilon = \frac{ST}{N^2}$ when $G = (\mathbb{Z}/N^c\mathbb{Z}, +)$ for $c \geq 2$. (the standard $\mathsf{3SUM\text{-}Indexing}$ problem) and $G = ((\mathbb{Z}/2\mathbb{Z})^{kn})$ for $k \geq 2$ (the $\mathsf{3XOR\text{-}Indexing}$ problem). If this conjecture is true, the previous theorem implies the existence of (exponentially strong) one-way functions in the random oracle model as long the preprocessing satisfies $S \leq N^{2-\delta}$ for $\delta > 0$. As per the discussion below Definition 11, Theorem 14 is vacuous in the regime where $S = \widetilde{\Omega}(N^2)$.

Open Questions

Our work raises a whole host of open problems. In terms of upper bounds, we showed how to refute the strong $\mathsf{3SUM\text{-}Indexing}$ conjecture of [GKLP17] using techniques from space-time tradeoffs for function inversion [Hel80, FN00], specifically the general function inversion algorithm of Fiat and Naor [FN00]. Can we improve the algorithm using the structure of the $\mathsf{3SUM\text{-}Indexing}$ problem? More ambitiously, can we refute the conjecture of Demaine and Vadhan [DV01]?

In terms of lower bounds, the natural remaining question is to extend our lower bound to adaptive strategies that potentially err with some probability.

Our work also opens the door to a new way to immunize cryptographic algorithms against backdoors, by leveraging on hardness results from the area of static data structures. Can we translate other (conjectured) static data structure lower bounds into backdoor-immunization schemes?

Acknowledgments

Many thanks to Erik Demaine for sending us a manuscript of his one-query lower bound with Salil Vadhan [DV01]. We also thank Henry Corrigan-Gibbs and Dima Kogan for discussions on this topic.

The work of AG is supported by a Rabin Postdoctoral Fellowship. The work of TH is supported in part by the National Science Foundation under grants CAREER IIS-1149662, CNS-1237235 and CCF-1763299, by the Office of Naval Research under grants YIP N00014-14-1-0485 and N00014-17-1-2131, and by a Google Research Award.

References

[AAD+12] Oswin Aichholzer, Franz Aurenhammer, Erik D. Demaine, Ferran Hurtado, Pedro Ramos, and Jorge Urrutia. On k-convex polygons. *Comput. Geom.*, 45(3):73–87, 2012.

[ACH+98] Esther M. Arkin, Yi-Jen Chiang, Martin Held, Joseph S. B. Mitchell, Vera Sacristan, Steven S. Skiena, and Tae-Cheon Yang. On minimum-area hulls. *Algorithmica*, 21(1):119–136, 1998.

[ACLL14] Amihood Amir, Timothy M. Chan, Moshe Lewenstein, and Noa Lewenstein. On hardness of jumbled indexing. In *ICALP 2014*, pages 114–125. Springer, 2014.

[AEK05] Daniel Archambault, Willam Evans, and David Kirkpatrick. Computing the set of all the distant horizons of a terrain. *Int. J. Comput. Geom. Appl.*, 15(06):547–563, 2005.

[AHI+01] Manuel Abellanas, Ferran Hurtado, Christian Icking, Rolf Klein, Elmar Langetepe, Lihong Ma, Belén Palop, and Vera Sacristán. Smallest color-spanning objects. In *ESA 2001*, pages 278–289. Springer, 2001.

[AHP08] Boris Aronov and Sariel Har-Peled. On approximating the depth and related problems. *SIAM J. Comput.*, 38(3):899–921, 2008.

[AKL+16] Amihood Amir, Tsvi Kopelowitz, Avivit Levy, Seth Pettie, Ely Porat, and B. Riva Shalom. Mind the gap: Essentially optimal algorithms for online dictionary matching with one gap. In *ISAAC 2016*, pages 12:1–12:12, 2016.

[AVW14] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *FOCS 2014*, pages 434–443. IEEE, 2014.

[AVWW14] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *ICALP 2014*, pages 39–51. Springer, 2014.

[BBS06] Elad Barkan, Eli Biham, and Adi Shamir. Rigorous bounds on cryptanalytic time/memory tradeoffs. In *CRYPTO 2006*, pages 1–21. Springer, 2006.

[BCC+13] Ayelet Butman, Peter Clifford, Raphaël Clifford, Markus Jalsenius, Noa Lewenstein, Benny Porat, Ely Porat, and Benjamin Sach. Pattern matching under polynomial transformation. *SIAM J. Comput.*, 42(2):611–633, 2013.

[BDP08] Ilya Baran, Erik D. Demaine, and Mihai Pătraşcu. Subquadratic algorithms for 3SUM. *Algorithmica*, 50(4):584–596, 2008.

[BFM18] Balthazar Bauer, Pooya Farshim, and Sogol Mazaheri. Combiners for backdoored random oracles. In *CRYPTO 2018*, pages 272–302. Springer, 2018.

[BHP01] Gill Barequet and Sariel Har-Peled. Polygon containment and translational min-hausdorff-distance between segment sets are 3SUM-hard. *Int. J. Comput. Geom. Appl.*, 11(04):465–474, 2001.

[BVKT98] Prosenjit Bose, Marc Van Kreveld, and Godfried Toussaint. Filling polyhedral molds. *Comput.-Aided Des.*, 30(4):245–254, 1998.

[BW09] Nikhil Bansal and Ryan Williams. Regularity lemmas and combinatorial algorithms. In *FOCS 2009*, pages 745–754. IEEE, 2009.

[CCI+19] Sergio Cabello, Jean Cardinal, John Iacono, Stefan Langerman, Pat Morin, and Aurélien Ooms. Encoding 3SUM. *arXiv:1903.02645*, 2019.

[CDGS18] Sandro Coretti, Yevgeniy Dodis, Siyao Guo, and John P. Steinberger. Random oracles and non-uniformity. In *EUROCRYPT 2018*, pages 227–258. Springer, 2018.

[CEHP07] Otfried Cheong, Alon Efrat, and Sariel Har-Peled. Finding a guard that sees most and a shop that sells most. *Discrete Comput. Geom.*, 37(4):545–563, 2007.

[CGI+16] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *ITCS 2016*, pages 261–270. ACM, 2016.

[CGK18] Henry Corrigan-Gibbs and Dmitry Kogan. The function-inversion problem: Barriers and opportunities. *ECCC TR18-182*, 2018.

[Cha18] Timothy M. Chan. More logarithmic-factor speedups for 3SUM, (median, +)-convolution, and some geometric 3SUM-hard problems. In *SODA 2018*, pages 881–897. SIAM, 2018.

[CHC09] Kuan-Yu Chen, Ping-Hui Hsu, and Kun-Mao Chao. Approximate matching for run-length encoded strings is 3SUM-hard. In *CPM 2009*, pages 168–179. Springer, 2009.

[CL15] Timothy M. Chan and Moshe Lewenstein. Clustered integer 3SUM via additive combinatorics. In *STOC 2015*, pages 31–40. ACM, 2015.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms.* MIT press, 2009.

[CMG+16] Stephen Checkoway, Jacob Maskiewicz, Christina Garman, Joshua Fried, Shaanan Cohney, Matthew Green, Nadia Heninger, Ralf-Philipp Weinmann, Eric Rescorla, and Hovav Shacham. A systematic analysis of the juniper dual EC incident. In *CCS 2016*, pages 468–479. ACM, 2016.

[CNE+14] Stephen Checkoway, Ruben Niederhagen, Adam Everspaugh, Matthew Green, Tanja Lange, Thomas Ristenpart, Daniel J. Bernstein, Jake Maskiewicz, Hovav Shacham, and Matthew Fredrikson. On the practical exploitability of dual EC in TLS implementations. In *USENIX 2014*, pages 319–335, 2014.

[dBdGO97] Mark de Berg, Marko M. de Groot, and Mark H. Overmars. Perfect binary space partitions. *Comput. Geom.*, 7(1-2):81–91, 1997.

[DGG+15] Yevgeniy Dodis, Chaya Ganesh, Alexander Golovnev, Ari Juels, and Thomas Ristenpart. A formal treatment of backdoored pseudorandom generators. In *EUROCRYPT 2015*, pages 101–126. Springer, 2015.

[DGK17] Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In *EUROCRYPT 2017*, pages 473–495. Springer, 2017.

[DGW19] Zeev Dvir, Alexander Golovnev, and Omri Weinstein. Static data structure lower bounds imply rigidity. In *STOC 2019*. ACM, 2019.

[DTT10] Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and PRGs. In *CRYPTO 2010*, pages 649–665. Springer, 2010.

[DV01] Erik D. Demaine and Salil P. Vadhan. Some notes on 3SUM, December 2001. Unpublished manuscript.

[EHPM06] Jeff Erickson, Sariel Har-Peled, and David M. Mount. On the least median square problem. *Discrete Comput. Geom.*, 36(4):593–607, 2006.

[Eri99] Jeff Erickson. Bounds for linear satisfiability problems. *Chicago J. Theor. Comput. Sci.*, 1999.

[FJM18] Marc Fischlin, Christian Janson, and Sogol Mazaheri. Backdoored hash functions: Immunizing HMAC and HKDF. In *CSF 2018*, pages 105–118. IEEE, 2018.

[FN00] Amos Fiat and Moni Naor. Rigorous time/space trade-offs for inverting functions. *SIAM J. Comput.*, 29(3):790–803, 2000.

[Fre17] Ari Freund. Improved subquadratic 3SUM. *Algorithmica*, 77(2):440–458, 2017.

[GKLP16] Isaac Goldstein, Tsvi Kopelowitz, Moshe Lewenstein, and Ely Porat. How hard is it to find (honest) witnesses? In *ESA 2016*, pages 45:1–45:16, 2016.

[GKLP17] Isaac Goldstein, Tsvi Kopelowitz, Moshe Lewenstein, and Ely Porat. Conditional lower bounds for space/time tradeoffs. In *WADS 2017*, pages 421–436. Springer, 2017.

[GLP17] Isaac Goldstein, Moshe Lewenstein, and Ely Porat. Orthogonal vectors indexing. In *ISAAC 2017*, pages 40:1–40:12, 2017.

[GO95] Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom.*, 5(3):165–185, 1995.

[Gol01] Oded Goldreich. *Foundations of Cryptography*, volume I. Basic tools. Cambridge University Press, 2001.

[GP14] Allan Grønlund and Seth Pettie. Threesomes, degenerates, and love triangles. In *FOCS 2014*, pages 621–630. IEEE, 2014.

[Gre13] Matthew Green. A few more notes on NSA random number generators, 2013.

[GT00] Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *FOCS 2000*, pages 305–313. IEEE, 2000.

[Hel80] Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Trans. Inf. Theory*, 26(4):401–406, 1980.

[HKNS15] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *STOC 2015*, pages 21–30. ACM, 2015.

[Imp96] Russell Impagliazzo. Very strong one-way functions and pseudo-random generators exist relative to a random oracle, January 1996. Unpublished manuscript.

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *STOC 1989*, pages 44–61. ACM, 1989.

[JV16] Zahra Jafargholi and Emanuele Viola. 3SUM, 3XOR, Triangles. *Algorithmica*, 74(1):326–343, 2016.

[KLM18] Daniel M Kane, Shachar Lovett, and Shay Moran. Near-optimal linear decision trees for k-SUM and related problems. In *STOC 2018*, pages 554–563. ACM, 2018.

[KP19] Tsvi Kopelowitz and Ely Porat. Personal communication, 2019.

[KPP16] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In *SODA 2016*, pages 1272–1287. SIAM, 2016.

[Lar12] Kasper Green Larsen. Higher cell probe lower bounds for evaluating polynomials. In *FOCS 2012*, pages 293–301. IEEE, 2012.

[LVWWW16] Andrea Lincoln, Virginia Vassilevska Williams, Joshua R. Wang, and R. Ryan Williams. Deterministic time-space trade-offs for k-SUM. In *ICALP 2016*, 2016.

[Păt10] Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *STOC 2010*, pages 603–610. ACM, 2010.

[Păt11] Mihai Pătraşcu. Unifying the landscape of cell-probe lower bounds. *SIAM J. Comput.*, 40(3):827–847, 2011.

[PTW10] Rina Panigrahy, Kunal Talwar, and Udi Wieder. Lower bounds on near neighbor search via metric expansion. In *FOCS 2010*, pages 805–814. IEEE, 2010.

[RTYZ18] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Correcting subverted random oracles. In *CRYPTO 2018*, pages 241–271. Springer, 2018.

[SEO03] Michael Soss, Jeff Erickson, and Mark Overmars. Preprocessing chains for fast dihedral rotations is hard or even impossible. *Comput. Geom.*, 26(3):235–246, 2003.

[Sie04] Alan Siegel. On universal classes of extremely random constant-time hash functions. *SIAM J. Comput.*, 33(3):505–543, 2004.

[Unr07] Dominique Unruh. Random oracles and auxiliary input. In *CRYPTO 2007*, pages 205–223. Springer, 2007.

[VW09] Virginia Vassilevska and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *STOC 2009*, pages 455–464. ACM, 2009.

[VW15] Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *IPEC 2015*, 2015.

[VW18] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *ICM 2018*, 2018.

[Wan14] Joshua R. Wang. Space-efficient randomized algorithms for k-sum. In *ESA 2014*, pages 810–829. Springer, 2014.

[Wil05] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005.

[WW10] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *STOC 2010*, pages 645–654. IEEE, 2010.

[YY96a] Adam L. Young and Moti Yung. Cryptovirology: Extortion-based security threats and countermeasures. In *S&P*, pages 129–140. IEEE, 1996.

[YY96b] Adam L. Young and Moti Yung. The dark side of "black-box" cryptography, or: Should we trust capstone? In *CRYPTO*, pages 89–103. Springer, 1996.

[YY97] Adam L. Young and Moti Yung. Kleptography: Using cryptography against cryptography. In *EUROCRYPT 1997*, pages 62–74. Springer, 1997.

[Zim98] Marius Zimand. Efficient privatization of random bits. In *MFCS 1998, Workshop "Randomized Algorithms"*, 1998.