No exponential quantum speedup for SIS$^\infty$ anymore

Robin Kothari* Ryan O'Donnell† Kewen Wu‡

Abstract

In 2021, Chen, Liu, and Zhandry presented an efficient quantum algorithm for the average-case ℓ_∞-Short Integer Solution (SIS$^\infty$) problem, in a parameter range outside the normal range of cryptographic interest, but still with no known efficient classical algorithm. This was particularly exciting since SIS$^\infty$ is a simple problem without structure, and their algorithmic techniques were different from those used in prior exponential quantum speedups.

We present efficient classical algorithms for all of the SIS$^\infty$ and (more general) Constrained Integer Solution problems studied in their paper, showing there is no exponential quantum speedup anymore.

*Google Quantum AI. Email: robin@robinkothari.com

†Computer Science Department, Carnegie Mellon University. Email: odonnell@cs.cmu.edu. Part of this work was done while consulting for Google Quantum AI.

‡Institute for Advanced Study. Email: shlw_kevin@hotmail.com. Part of the work was done while at Google.

Contents

1 Introduction

Finding new problems where quantum algorithms yield an exponential speedup over classical ones remains a central challenge in quantum computer science. We have long known a few classes of problems with a likely exponential quantum speedup, such as those based on the simulation of quantum systems or based on the hidden subgroup problem (e.g., integer factorization, discrete log). Beyond those, it is rare and exciting to find a genuinely new class of problems with potential exponential quantum speedup, particularly if the problems are natural and previously studied.

In 2021, Chen, Liu, and Zhandry [CLZ22] did precisely this. They presented an efficient quantum algorithm for the well-known *Short Integer Solution in ℓ_∞ norm* (SIS^∞) problem, in a less-studied parameter regime, but one with no known classical polynomial-time algorithm. Here is an example result they proved (see Theorem 1.4 for more). Let q be a prime.[1]

> **Sample CLZ theorem.** *There is a* $\mathrm{poly}(m, q)$*-time* quantum *algorithm that, given a* uniformly random $H \in \mathbb{F}_q^{n\times m}$ *where*[2] $m \geq Cq^4 \log q \cdot n^3$*, finds a "short" nonzero* $x \in \mathbb{F}_q^m$ *satisfying* $Hx = 0$*, where "short" means* $\|x\|_\infty < \lfloor q/2 \rfloor$*, i.e.,* $x_i \neq \pm\lfloor q/2 \rfloor$ *for all* i*.*

The particularly exciting part about this work is that it used genuinely different techniques from existing quantum speedups. Furthermore, in a more standard and more challenging parameter regime — e.g., $m = Cn$ and "short" meaning[3] $\|x\|_\infty \leq q/4$ — the SIS^∞ problem is of significant cryptographic interest. Variants of the problem underlie the security of several (candidate post-quantum) cryptographic systems, such as Dilithium [DKL+18] and Wave [DST19].

The Chen–Liu–Zhandry quantum algorithm is based on a quantum reduction by Regev [Reg09], who based the *hardness* of the Learning With Errors problem on the hardness of finding short lattice vectors. A similar reduction shows that SIS^∞ can be solved efficiently if a certain decoding problem can be solved efficiently. Instead of using Regev's reduction to show hardness, Chen, Liu, and Zhandry devise a clever, efficient quantum algorithm for the decoding problem (in a certain parameter regime), and thereby obtain an efficient quantum algorithm for SIS^∞.

These *algorithmic* applications of Regev's reduction have since attracted considerable interest [CLZ22, CT23, YZ24, JSW+24, CT24]. Besides the algorithm for SIS^∞, two other quantum algorithms that use the same primitive are: (1) the algorithm of Yamakawa and Zhandry [YZ24] that achieves an exponential quantum–classical black-box separation for a search problem relative to a random oracle; and, (2) the Decoded Quantum Interferometry (DQI) algorithm for various optimization problems, such as the Optimal Polynomial Intersection (OPI) problem [JSW+24, CT24]. The result by Yamakawa and Zhandry can be made non-oracular by instantiating the random oracle with a cryptographic hash function, providing a concrete problem that presumably retains the exponential quantum speedup. This leaves us with at least three non-oracular problems based on this algorithmic primitive with potential exponential speedups: SIS^∞, OPI, and Yamakawa–Zhandry.

Our main result is to dequantize[4] [CLZ22]. We show that SIS^∞, and a further generalization they study, can be solved by a classical algorithm whose efficiency even outperforms that of their quantum algorithm. For example, we prove (see Theorem 1.5 for more) the following theorem.

> **Sample new theorem.** *There is a* $\mathrm{poly}(m, \log q)$*-time* classical *algorithm that, given* any $H \in \mathbb{F}_q^{n\times m}$ *where* $m \geq Cn^3$*, finds a "short" nonzero* $x \in \mathbb{F}_q^m$ *satisfying* $Hx = 0$*,*

[1] We use q (instead of p) to be consistent with cryptography literature. Our results also work for prime power or composite modulus. See Section 5 for details.

[2] For simplicity, we use C to hide constants that may be different from place to place and may depend on other parameters described as "constant". In later sections we will formally state these constant dependencies.

[3] Here x's entries are interpreted in $\{-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor\}$.

[4] We do not simulate the CLZ algorithm in a classical way. Rather, we provide a classical algorithm directly.

where "short" means $\|x\|_\infty \le \lfloor q/6 \rfloor$*, i.e.,* $-\lfloor q/6 \rfloor \le x_i \le \lfloor q/6 \rfloor$ *for all* i*.*

We highlight several advantages of our result: (1) it is classical and deterministic; (2) it works for worst-case H; (3) the requirement on m has no dependence on q; (4) it is $\mathrm{poly}(n)$-time even when $q = 2^{\mathrm{poly}(n)}$; and (5) the notion of "short" is much stricter. (Our algorithm's classical running time is even faster than CLZ's quantum running time in the setting of comparable m; see Section 5.) As we also discuss in Section 5, for this particular sample theorem, a similar but slightly weaker version (without features (3)(4)) can be extracted from a recent work of Imran and Ivanyos [II24].

In general, the main theorem in this work gives a classical algorithm that outperforms the most general quantum algorithm from [CLZ22]. This quantum algorithm is for a generalization of SIS^∞ we call Constrained Integer Solution (CIS) problem, in which the entries of x are restricted to lie in a general set A, as opposed to an interval. (For more details, see Section 1.4.)

Theorem. ([CLZ22])**.** *Let* $k \ge 2$ *be a constant and* $A \subseteq \mathbb{F}_q$ *be of size* $|A| = q - k + 1$*. There is a* $\mathrm{poly}(m, q)$*-time quantum algorithm that, given a uniformly random* $H \in \mathbb{F}_q^{n \times m}$ *where*

$$m \ge C q^4 \log q \cdot n^k, \tag{1}$$

finds a nonzero $x \in A^m$ *satisfying* $Hx = 0$*.*

Our main contribution is a classical algorithm that improves their quantum algorithm. Note that we not only improve their q-dependence in all cases, but improve their n-dependence in most cases.

Theorem. (Main)**.** *There is a* $\mathrm{poly}(m, q)$*-time classical algorithm for the above problem, even for*

$$m \ge C \log q \cdot \begin{cases} n^2 & \text{whenever } q > 4^{k-1} \\ n^{k-1} & \text{whenever } k \ge 3 \text{ and } q \ge 7 \\ n^k & \text{in general for all } k \ge 2 \text{ and } q \ge 3. \end{cases}$$

1.1 A first warm-up: $\mathbb{F}_3^n$-Subset-Sum

The general SIS^∞ and CIS problems have several parameters, so we will warm up to a full description of the problem with some special cases. The simplest special case is $\mathbb{F}_3^n$-Subset-Sum, which was already considered to have a potential exponential quantum speedup in [CLZ22].

The $\mathbb{F}_3^n$-Subset-Sum problem is just a vector version of the classic Subset-Sum problem, but over $\mathbb{F}_3^n$ rather than the integers, and with the target fixed to 0:

Given m vectors $h_1, \ldots, h_m \in \mathbb{F}_3^n$, find a nonempty subset $S \subseteq [m]$[5] such that $\sum_{i \in S} h_i = 0$.

In general, we can consider $\mathbb{F}_q^n$-Subset-Sum for any q, and $q = 3$ is the first interesting case.

The reader familiar with SIS^∞ will recognize $\mathbb{F}_3^n$-Subset-Sum as a worst-case version of it, in which $q = 3$ and a vector $x \in \mathbb{F}_3^m$ is considered "short" if it is in $\{0, 1\}^m$. Indeed, as we will discuss in Section 5.4, $\mathbb{F}_3^n$-Subset-Sum is (roughly) equivalent to a wide variety of problems: "Binary-Error LWE with $q = 3$", "list-recovery for $\mathbb{F}_3$-linear codes", "ternary syndrome decoding with large (maximal) weight", "LIN-SAT over $\mathbb{F}_3$", "Learning From Disequations over $\mathbb{F}_3$", and more.

As posed, the $\mathbb{F}_3^n$-Subset-Sum problem might not always have a solution, particularly if m is small compared to n. But it is a mathematically nontrivial fact (discussed in Section 5.5) that once

[5] For any integer $n \ge 1$, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$.

m crosses the sharp threshold $2n$, a solution *always* exists. We will only study the problem when $m > 2n$, so there will always be a solution to find.

$\mathbb{F}_3^n$-Subset-Sum is also very interesting in the average-case setting, where the input vectors are chosen uniformly at random and the goal is to succeed on almost all inputs. Here it simple to show that a solution exists with overwhelming probability beyond threshold $(\log_2 3)n \approx 1.58n$, a slightly smaller threshold than the one for worst-case existence. It is this average-case version that is considered by Chen, Liu, and Zhandry. Although they were mainly interested in SIS$^\infty$ with larger q, their work also contained the following interesting result.

Theorem 1.1 (Special case of [CLZ22, Remark 4])**.** *The $\mathbb{F}_3^n$-Subset-Sum problem with uniformly random input vectors can be solved by a bounded-error* poly(m)*-time quantum algorithm provided $m \geq Cn^2$.*[6]

However, this problem can be solved efficiently and deterministically by a classical algorithm, even in the worst case. This was already shown implicitly in a 2007 paper of Ivanyos, Sanselme, and Santha: [ISS07, Claim 1] implies that $\mathbb{F}_3^n$-Subset-Sum can be solved for worst-case inputs when $m \geq (n+1)(n+2)/2 \sim n^2/2$. We present an improved classical algorithm.

Theorem 1.2 (Proved in Section 2)**.** *The worst-case $\mathbb{F}_3^n$-Subset-Sum problem can be solved by a deterministic* poly(m)*-time classical algorithm provided $m \geq n^2/3 + Cn \log n$.*

1.2 Short Integer Solution in ℓ_∞ norm, SIS$^\infty$

We now discuss the main SIS$^\infty$ problem studied by Chen, Liu, and Zhandry. Given a set of m vectors over $\mathbb{F}_q^n$, the goal is to find a nontrivial linear combination of these that sums to 0, where the vector x of coefficients should have small infinity norm, meaning $x \in \{-s, \ldots, s\}^m$ for some positive integer s. Throughout this paper, we work with odd prime field $\mathbb{F}_q$ and view it as the set $\{-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor\}$. Thus the largest infinity norm of vectors in $\mathbb{F}_q^m$ is $\lfloor q/2 \rfloor$.

Definition 1.3. In the SIS$^\infty(n, m, q, s)$ problem for a prime $q > 3$, we are given a matrix $H \in \mathbb{F}_q^{n \times m}$ and an integer $s \in \{1, \ldots, \lfloor q/2 \rfloor\}$. The goal is to find a nonzero vector $x \in \mathbb{F}_q^m$ with $\|x\|_\infty \leq s$ satisfying $Hx = 0$.

This problem gets easier as m gets larger and as s gets closer to $\lfloor q/2 \rfloor$. For example, when $s = \lfloor q/2 \rfloor$, every nonzero vector is allowed and we only need $m = n + 1$ vectors.

This problem is interesting both in the worst case and the average case.[7] The main result of Chen, Liu, and Zhandry is a quantum algorithm for the problem in the average case, where the matrix H is drawn uniformly at random from $\mathbb{F}_q^{n \times m}$.

Theorem 1.4 ([CLZ22, Theorem 2])**.** *For any odd constant $k \geq 3$, there is a bounded-error* poly(m, q)*-time quantum algorithm for* average-case SIS$^\infty$ *when*

$$s = \frac{q-k}{2} \quad \text{and} \quad m \geq Cq^4 \log q \cdot n^k. \tag{2}$$

This algorithm works for average-case inputs and is only poly(n)-time when $q \leq$ poly(n). The main result in our work is a classical algorithm that works for worst-case inputs, achieves a significantly smaller s, and allows q to be exponentially large in n.

[6] C is not explicit in [CLZ22]. However, we can show (proof omitted) that $m \geq (n+1)(n+2)/2$ vectors suffice. We also remark that certain conjectures would imply $0 < C < 1/2$; but these are based on a generalization of the Arora–Ge algorithm [AG11] without rigorous performance guarantees; see, e.g., [Ste24].

[7] Assuming worst-case hardness of some lattice problems and passing through the SIS problem in ℓ_2 norm [Ajt96, MR07], the average-case SIS$^\infty$ problem is hard when $s = O(1)$, $q = \Theta(n^2 \log n)$, and $m = \Theta(n \log n)$.

Theorem 1.5 (Proved in Section 4.4). *For any constant $k \geq 2$, there is a deterministic* $\mathrm{poly}(m, \log q)$*-time classical algorithm for* worst-case SIS^∞ *when*

$$s = \left\lfloor \frac{q}{2k} \right\rfloor \quad \textit{and} \quad m \geq Cn^k. \tag{3}$$

We also have a variant of Theorem 1.5 which has an additional assumption on k, but a substantially simpler proof and better control on the constants.

Theorem 1.6 (Simpler version; proved in Section 3.1). *For any $k \leq \lfloor q/2 \rfloor$ that is a positive-integer power of 2, there is a deterministic* $\mathrm{poly}(m, \log q)$*-time classical algorithm for* worst-case SIS^∞ *when*

$$s = \left\lfloor \frac{q}{2k} \right\rfloor \quad \textit{and} \quad m \geq \left(\frac{n+1}{c_q} \right)^k, \tag{4}$$

where $c_q = \sqrt{2 + \frac{2}{q-1} - o(1)} \geq 1$.

We present the proof of this weaker theorem in Section 3, as a warm-up to the more elaborate proof needed for Theorem 1.5. As we discuss in Section 5, this easier proof is based on a method of Imran and Ivanyos [II24], but improves upon it by allowing exponentially large q.

1.3 $\mathbb{F}_q^n$-Subset-Sum

It is natural also to study the $\mathbb{F}_q^n$-Subset-Sum problem, which is like SIS^∞ with $s = 1$, but even harder: rather than seeking a nonzero $x \in \{-1, 0, 1\}^m$ with $Hx = 0$, we seek $x \in \{0, 1\}^m$. In order to achieve $m = \mathrm{poly}(n)$, it seems necessary to focus on constant q. We also exclude $q = 3$, which is already studied in Section 1.1.

There are two prior algorithms we can compare against: a quantum algorithm by Chen, Liu, and Zhandry [CLZ22] and a classical one by Imran and Ivanyos [II24].

Theorem 1.7 (Follows from [CLZ22, Remark 4]). *For constant prime $q > 3$, the* $\mathbb{F}_q^n$-Subset-Sum *problem with m uniformly random input vectors can be solved by a bounded-error* $\mathrm{poly}(m)$*-time quantum algorithm provided $m \geq Cn^{q-1}$.*

Theorem 1.8 (Follows from [II24, Remark 4]). *For constant prime $q > 3$, the* $\mathbb{F}_q^n$-Subset-Sum *problem with m* worst-case *input vectors can be solved by a deterministic* $\mathrm{poly}(m)$*-time classical algorithm provided $m \geq Cn^{\frac{1}{4}\overline{q}\log_2 \overline{q}}$, where $\overline{q}$ is q rounded up to the next integer power of 2.*

Note that the exponent here is $\Omega(q \log q)$, which exceeds $q - 1$ for $q \neq 7$.

We prove the following result for $\mathbb{F}_q^n$-Subset-Sum, which strictly improves Theorem 1.7. Note that our result is incomparable with Theorem 1.8, as we have a noticeably better exponent, but works in the average case.

Theorem 1.9 (Proved in Section 4.5). *For constant prime $q > 3$, the* $\mathbb{F}_q^n$-Subset-Sum *problem with m uniformly random input vectors can be solved by a deterministic* $\mathrm{poly}(m)$*-time classical algorithm provided $m \geq Cn^{\lceil\lceil q/2\rceil\rceil}$, where $\lceil\lceil q/2\rceil\rceil$ denotes the smallest even integer exceeding $q/2$.*

Note that the exponent is $q/2 + o(1)$, which is always at most $q - 1$ and strictly so for $q \neq 5$.

Once again, the proof of Theorem 1.9 is somewhat involved, and we present a less complicated version as Theorem 3.12, for $m \geq Cn^{\overline{q-1}}$, where $\overline{q-1}$ is $q - 1$ rounded up to the next power of 2.

1.4 Constrained Integer Solution (CIS)

While the main result presented by Chen, Liu, and Zhandry is about SIS^∞, their algorithm works for a more general problem, as pointed out in [CLZ22, Remark 4]. In this version, instead of requiring that the solution x has small infinity norm, we require that each entry of x belongs to a specified allowed set $A \subseteq \mathbb{F}_q$. We call this problem "Constrained Integer Solution (CIS)" or sometimes A-CIS to emphasize the set A.

Note that setting $A = \{-s, \ldots, s\}$ recovers the SIS^∞ problem; and setting $A = \{0, 1\}$ recovers the Subset-Sum problem. We also describe how CIS unifies Yamakawa-Zhandry [YZ24] and OPI [JSW+24] in Section 5.

Definition 1.10. In the $\mathrm{CIS}(n, m, q, A)$ problem for a prime $q \geq 3$, we are given a matrix $H \in \mathbb{F}_q^{n \times m}$ and a set $A \subseteq \mathbb{F}_q$ of size $2 \leq |A| \leq q - 1$. The goal is to find a nonzero vector $x \in A^m$ satisfying $Hx = 0$.

Like [CLZ22], we will only study this problem in the average case, where H is a uniformly random matrix in $\mathbb{F}_q^{n \times m}$, as the worst-case instances may not guarantee solutions. We also remark that the problem seems quite hard unless $|A|$ is close to q; thus we parameterize $|A| = q - k + 1$.

It turns out that the same quantum algorithm in [CLZ22] that solves SIS^∞ can be used for CIS with similar bounds.

Theorem 1.11 ([CLZ22, Remark 4])**.** *Let $k \geq 2$ be a constant and $A \subseteq \mathbb{F}_q$ be of size $|A| = q-k+1$. There is a bounded-error $\mathrm{poly}(m, q)$-time quantum algorithm for average-case A-CIS when*

$$m \geq Cq^4 \log q \cdot n^k. \tag{5}$$

We give classical algorithms for the same problem. First, we have a simple argument to handle the case when q is much larger than k. This is in fact the motivating setting in [CLZ22], as they typically describe q being polynomial in n and k being constant.

Theorem 1.12 (Proved in Section 3.3)**.** *Let $k \geq 2$ be a constant and $A \subseteq \mathbb{F}_q$ be of size $|A| = q-k+1$. Assume $q > 4^{k-1}$. There is a deterministic $\mathrm{poly}(m, q)$-time classical algorithm for average-case A-CIS when*

$$m \geq C \log q \cdot n^2. \tag{6}$$

Note here that the exponent on n is fixed to 2; it does not grow with k as in Theorem 1.11.

On the other hand, for general q, we have another, more involved, classical algorithm. Compared with Theorem 1.11, our requirement on m always has much better q-dependence and has better n-dependence in almost all settings.

Theorem 1.13 (Proved in Section 4.6)**.** *For any constant $k \geq 3$ and prime $q > 5$, given a set $A \subseteq \mathbb{F}_q$ with $|A| = q - k + 1$, there is a deterministic $\mathrm{poly}(m, q)$-time classical algorithm for average-case A-CIS with*

$$m \geq C \log q \cdot n^{k-1}. \tag{7}$$

The above bound also holds when $k = 3$ and $q = 5$.

In the remaining cases of $k = 2$ or $k = 4, q = 5$, the algorithm requires $m \geq C \log q \cdot n^k$.

Our focus is to provide *some* bound that matches or improves the known quantum algorithms; and we do not make the effort to exhaust the *best possible* saving in all parameter regimes.

1.5 Results summary

In Table 1, we compare efficient algorithms for SIS^∞-type problems. For simplicity, we use ϵ to hide $o(1)$ factors that go to zero as n or q gets larger. Footnote-sized conditions inside parentheses are additional assumptions of the problem or algorithm.

Table 1: Comparison of efficient algorithms for SIS^∞-type problems.

Problem \ Work	[CLZ22] **(quantum)**	[ISS07, II24] **(classical)**	**Present work (classical)**
A-CIS $\lvert A\rvert = q-k+1$ (constant k) (average-case)	$m \geq Cq^4 \log q \cdot n^k$		$m \geq C \log q \cdot n^2$ ($q > 4^{k-1}$) $m \geq C \log q \cdot n^k$ (general q)
$\mathbb{F}_3^n$-Subset-Sum	$m \geq Cn^2$ (average-case)	$m \geq (1/2+\epsilon)n^2$ (worst-case)	$m \geq (1/3+\epsilon)n^2$ (worst-case)
$\mathbb{F}_q^n$-Subset-Sum (constant q)	$m \geq Cn^{q-1}$ (average-case)	$m \geq n^{Cq \log q}$ (worst-case)	$m \geq Cn^{q/2+\epsilon}$ (average-case)
SIS^∞ (constant k)	$s = (q-k)/2$ $m \geq Cq^4 \log q \cdot n^k$ (odd k) (average-case)	$s = \lfloor q/(2k) \rfloor$ $m \geq q^{Ck \log k} \cdot n^k$ (k power of 2) (worst-case)	$s = \lfloor q/(2k) \rfloor$ $m \geq Cn^k$ (worst-case)

Paper organization. In Section 2, we present our algorithms on the $\mathbb{F}_3^n$-Subset-Sum problem. In Section 3, we describe a simple halving trick that improves upon the techniques in [II24] to reduce the weight of solutions, which leads to simple algorithms for SIS^∞, Subset-Sum, and CIS. Then in Section 4, we develop extra techniques to improve previous algorithms. Finally in Section 5, we mention further improvements on our results and discuss problems, algorithms, and motivations related to our work.

2 $\mathbb{F}_3^n$-Subset-Sum algorithms

The goal of this section is to handle the simplest field $\mathbb{F}_3 = \{-1, 0, 1\}$ and prove Theorem 1.2.

We start with a weaker version of Theorem 1.2 with the simplest proof. Then we identify some room for improvement and finally obtain Theorem 1.2. We will use zero-sum to refer to a linear combination of vectors that equals zero; and we say it is nontrivial if it uses some nonzero coefficient in the linear combination.

Theorem 2.1. *The worst-case* $\mathbb{F}_3^n$-Subset-Sum *problem can be solved in deterministic* $\mathrm{poly}(m)$ *time when* $m \geq (n+1)^2$.

Proof. Take the set S_1 of the first $n+1$ vectors and find a nontrivial linear combination of them equaling 0. Collecting like coefficients together, this yields $u_1 - u_1' = 0$, where u_1 and u_1' are vectors

formed by disjoint subset-sums from S_1 and they are not both empty. Note that $u_1+u_1' = u_1+u_1 = -u_1$, so $-u_1$ is also a subset-sum from S_1.

Repeat for subsequent sets of $n+1$ vectors, $S_2, \dots, S_{n+1}$, producing $u_2, \dots, u_{n+1}$. For each $i \in [n+1]$, we have that $\pm u_i$ is a subset-sum from S_i.

Finally, find a nontrivial linear combination of $u_1, \dots, u_{n+1}$ equaling 0. Collecting like coefficients, this yields $v - v' = 0$, where v and v' are subset-sums of the u_i's, not both empty. But now v is a subset-sum of the original vectors (since the u_i's are), and so too is $-v'$ (since the $-u_i$'s are). This gives a nontrivial zero-sum from the original vectors as desired. □

One way to improve Theorem 2.1 is to observe a dimension reduction: once u_1 is created, we can project later vectors onto the space orthogonal to u_1; this is because $0 \cdot u_1, 1 \cdot u_1, -1 \cdot u_1$ can all be replaced by a subset-sum from S_1. This leads to the following Theorem 2.2, which is a special case of [ISS07, Claim 1] and we present a short proof here.

Theorem 2.2. *The worst-case* $\mathbb{F}_3^n$-Subset-Sum *problem can be solved in deterministic* $\mathrm{poly}(m)$ *time when* $m \geq (n+1)(n+2)/2$.

Proof. It will be more convenient to work with the following more general statement: given $m \geq (\ell+1)(\ell+2)/2$ vectors in a linear subspace V of $\mathbb{F}_3^n$ with dimension ℓ, we can find in polynomial time a nontrivial zero-sum. Then Theorem 2.2 follows by setting $\ell = n$ and $V = \mathbb{F}_3^n$.

We prove the above general statement by induction on ℓ and the runtime will be clear from the analysis. The base case of $\ell = 0$ is trivial, as the given vector must be 0. For the induction step, let S be the set of first $\ell+1$ vectors and T the last $\ell(\ell+1)/2$. The vectors in S must be linearly dependent, so we can find a nontrivial linear combination of them that sums to 0. Write this linear combination as $u - u' = 0$, where u, u' are disjoint subset-sums from S and are not both empty. Similar to the proof of Theorem 2.1, $-u = u + u'$ is also a subset-sum from S.

Now if $u = 0$ then we are done. Otherwise write $V = \mathrm{span}\{u\} \oplus W$, where W has dimension $\ell - 1$. Each vector $v_i \in T$ can be written as $c_i u + w_i$ for some $c_i \in \mathbb{F}_3$ and $w_i \in W$. Applying induction to the w_i's gives a nontrivial zero-sum, i.e., a nonempty collection $\{w_i : i \in I \subseteq T\}$ with $\sum_{i \in I} w_i = 0$. Now the associated collection of original vectors $\{v_i : i \in I\}$ sums to $\sum_{i \in I} v_i = c \cdot u$ for $c = \sum_{i \in I} c_i \in \mathbb{F}_3$. By our construction of u above, $-c \cdot u$ can be expressed as a subset-sum from $S' \subseteq S$. This means $\sum_{i \in S' \cup I} v_i = \sum_{i \in I} v_i - c \cdot u = 0$ is a desired nontrivial zero-sum. □

To further optimize parameters, we observe that each time the dimension reduction relies on a nontrivial linear equation. While both Theorem 2.1 and Theorem 2.2 generate such equations by the simple rank+1 bound, we can go beyond this, thanks to the following result.

Lemma 2.3. *Let* $\ell \geq 0$ *be an integer and* $t = \lceil \log_3(\ell+1) \rceil$. *Given* $m \geq \ell + t$ *vectors in a linear subspace* V *of* $\mathbb{F}_3^n$ *with dimension* ℓ, *we can find in deterministic* $\mathrm{poly}(n, m)$ *time a nontrivial zero-sum that uses at most* $2(\ell+1)/3 + t$ *vectors.*

Proof. Without loss of generality we assume the input vectors span the full subspace V, since otherwise we can always work with the smaller subspace and fewer vectors. By taking an invertible linear transform, we assume the input vector are standard basis vectors $e_1, \dots, e_\ell$ and additional vectors $v_1, \dots, v_t$.

For each $\alpha = (\alpha_1, \dots, \alpha_t) \in \mathbb{F}_3^t$, define $v_\alpha = \alpha_1 v_1 + \cdots + \alpha_t v_t$. Now it suffices to find some $\alpha \neq 0^t$ such that v_α has at most $2(\ell+1)/3$ nonzero entries, in which case these nonzero entries can be canceled by $2(\ell+1)/3$ standard basis vectors to obtain the desired zero-sum.

To find such an α, we consider a uniform $\alpha \in \mathbb{F}_3^t$. Then the vector v_α has at most $2\ell/3$ nonzeros in expectation; this is because every coordinate of v_α is either constantly zero or uniform in $\mathbb{F}_3$.

Since the trivial choice $\alpha = 0^t$ happens with probability 3^{-t}, there is some $\alpha \neq 0^t$ such that v_α has at most

$$\frac{2\ell/3}{1-3^{-t}} \leq \frac{2\ell/3}{1-\frac{1}{\ell+1}} = \frac{2(\ell+1)}{3}$$

nonzeros and α can be efficiently found by brute-force enumeration. □

Now we are ready to prove Theorem 1.2.

Proof of Theorem 1.2. We follow the same strategy of the inductive arguments in Theorem 2.2. The only difference is we use Lemma 2.3 to find the set S of linear dependent vectors. The detailed calculation is as follows.

Let $m_0 = 1$ and

$$m_\ell = \max\left\{\ell, m_{\ell-1} + \frac{2(\ell+1)}{3}\right\} + \lceil \log_3(\ell+1) \rceil. \tag{8}$$

We will prove the statement: given $m \geq m_\ell$ vectors in a linear subspace V of $\mathbb{F}_3^n$ with dimension ℓ, we can find in polynomial time a nontrivial zero-sum. The base case $\ell = 0$ is again trivial. Now consider the induction step at dimension $\ell \geq 1$. Define $t = \lceil \log_3(\ell+1) \rceil$. Since we have $m \geq m_\ell \geq \ell + t$ vectors, we can use Lemma 2.3 to find a set of at most $m' = 2(\ell+1)/3 + t$ vectors with linear dependence; then apply induction hypothesis for dimension $\ell - 1$ with the remaining $m - m' \geq m_{\ell-1}$ vectors, similar to the proof of Theorem 2.2.

Finally, to analyze Equation (8), we define

$$\overline{m}_\ell = \frac{(\ell+1)(\ell+2)+1}{3} + \ell \cdot \lceil \log_3(\ell+1) \rceil.$$

Then a simple inductive argument proves $m_\ell \leq \overline{m}_\ell$: the base case $\ell = 0$ is trivial; the inductive case $\ell \geq 1$ is proved by observing $\ell + \lceil \log_3(\ell+1) \rceil \leq \ell \cdot \lceil \log_3(\ell+1) \rceil \leq \overline{m}_\ell$ and

$$m_{\ell-1} + \frac{2(\ell+1)}{3} + \lceil \log_3(\ell+1) \rceil \leq \overline{m}_{\ell-1} + \frac{2(\ell+1)}{3} + \lceil \log_3(\ell+1) \rceil \leq \overline{m}_\ell.$$

This completes the proof of Theorem 1.2 by setting $\ell = n$ and $V = \mathbb{F}_3^n$. □

3 A halving trick and its applications

The main feature we explore in Section 2 is to use linear dependence to construct a subset-sum $u \in \mathbb{F}_3^n$ such that $0 \cdot u, 1 \cdot u, -1 \cdot u$ can all be expressed as subset-sums. For larger fields and considering signs, this can be generalized as the following definition of *reducible* vector, which will be helpful in designing algorithms for SIS$^\infty$, Subset-Sum, and CIS.

The goal of this section is to give a primitive exposition of the halving trick using reducible vectors. We note that our halving trick is inspired by a similar one in [II24], and our main contribution here is to remove the dependence on the field size q in [II24].

Definition 3.1 (Bounded Vector Sum)**.** Let $v_1, \ldots, v_m$ be vectors and let $h \geq 1$ be an integer. We say vector u is a $(\pm h)$-sum of the v_i's if we can choose integer $-h \leq \alpha_i \leq h$ for each $i \in [m]$ such that $u = \sum_{i \in [m]} \alpha_i v_i$; and it is a $(\pm h)$-zero-sum if $u = 0$. We say the sum is nontrivial if α_i's are not all zero.

Definition 3.2 (Reducible Vector)**.** Let $v_1, \ldots, v_m$ be vectors and let $1 \leq h' < h$ be integers. We say that vector u is $(\pm h \to \pm h')$-reducible for the v_i's if

- u is a nontrivial (± 1)-sum of the v_i's;
- for every integer $-h \le c \le h$, the vector $c \cdot u$ is a $(\pm h')$-sum of the v_i's.

For clarity, our presentation in this section will focus on the existence of vector sums with certain properties (e.g., a solution to SIS^∞) and the runtime of such a construction will be obvious from the proof and thus omitted.

We begin with showing that zero-sums give reducible vectors.

Lemma 3.3. *Let $h > 1$ be an integer. Given a nontrivial $(\pm h)$-zero-sum of vectors $v_1, \ldots, v_m$,*

$$\alpha_1 v_1 + \cdots + \alpha_m v_m = 0 \quad \text{where each } -h \le \alpha_i \le h \text{ is an integer,} \tag{9}$$

we can find in deterministic $\mathrm{poly}(m, \log h)$ time a $(\pm h \to \pm\lfloor h/2 \rfloor)$-reducible u for the v_i's.

Proof. Define $a = \max_{i \in [m]} |\alpha_i|$, which satisfies $1 \le a \le h$ since Equation (9) is a nontrivial $(\pm h)$-zero-sum. Define $h' = \lfloor h/2 \rfloor$. We first handle the simpler case that $a > h'$. In this case, we construct u by

$$u = \sum_{i \in [m] \colon \alpha_i > h'} v_i - \sum_{i \in [m] \colon \alpha_i < -h'} v_i,$$

which is a nontrivial (± 1)-sum since $a > h'$. In addition for any $0 \le c \le h$, if $c \le h'$, then $c \cdot u$ is a $(\pm c)$-sum, which is immediately a $(\pm h')$-sum; if $h' < c \le h$, then we express $c \cdot u$ by

$$c \cdot u = c \cdot u - \sum_{i \in [m]} \alpha_i v_i = \sum_{i \colon |\alpha_i| \le h'} (-\alpha_i) v_i + \sum_{i \colon \alpha_i > h'} (c - \alpha_i) v_i + \sum_{i \colon \alpha_i < h'} -(c + \alpha_i) v_i,$$

which is a $(\pm h')$-sum since $|\alpha_i| \le h$ and $h' = \lfloor h/2 \rfloor$. Similar calculation works for $-h \le c \le 0$.

For the other case that $1 \le a \le h'$, we compute $t = \lfloor h/a \rfloor$ and consider the following equivalent form of Equation (9):

$$\alpha'_1 v_1 + \cdots + \alpha'_m v_m = 0 \quad \text{where } \alpha'_i = t \cdot \alpha_i \text{ satisfies } -h \le \alpha'_i \le h.$$

Since $1 \le a \le h' = \lfloor h/2 \rfloor$, we know $t = \lfloor h/a \rfloor > h/(2a)$ and hence $\max_{i \in [m]} |\alpha'_i| = t \cdot a > h/2 \ge h'$. Therefore this reduces to the above case.

Finally we remark that the procedure above is oblivious to the vectors $v_1, \ldots, v_m$; and it only works on the coefficients $\alpha_1, \ldots, \alpha_m$ to construct u and express each $c \cdot u$ as a $(\pm\lfloor h/2 \rfloor)$-sum. Therefore it has runtime $\mathrm{poly}(m, \log h)$ that is independent of n. □

Lemma 3.3 leads to the following corollary, which halves the weight of the SIS^∞ solutions at the expense of squaring.

Corollary 3.4. *Let q be a prime and let $h \ge 2$ be an integer. Suppose there is a deterministic algorithm $\mathcal{A}$, given m input vectors in $\mathbb{F}_q^n$ and running in time T, that outputs a nontrivial $(\pm h)$-zero-sum.*

Then there is a deterministic algorithm $\mathcal{B}$, given m^2 vectors in $\mathbb{F}_q^n$ and running in time $(m+1)T + \mathrm{poly}(m, \log h)$, that outputs a nontrivial $(\pm\lfloor h/2 \rfloor)$-zero-sum.

Proof. The algorithm $\mathcal{B}$ will split the input vectors $v_1, \ldots, v_{m^2}$ into m batches $S_1, \ldots, S_m$ of size m. For each $i \in [m]$, $\mathcal{B}$ applies $\mathcal{A}$ to the vectors in S_i to get a nontrivial $(\pm h)$-zero-sum from S_i, and then applies Lemma 3.3 to get a $(\pm h \to \pm\lfloor h/2 \rfloor)$-reducible $u^{(i)}$ for S_i. Note that if any $u^{(i)} = 0$, then we are already done by outputting $u^{(i)}$.

Assume $u^{(1)}, \dots, u^{(m)}$ are all nonzero vectors. Then $\mathcal{B}$ applies $\mathcal{A}$ again, to $u^{(1)}, \dots, u^{(m)}$, thereby getting a nontrivial $(\pm h)$-zero-sum $\sum_i c_i u^{(i)} = 0$, where the c_i's are not all zero. Since each $u^{(i)}$ is $(\pm h \to \pm\lfloor h/2 \rfloor)$-reducible, we can replace $c_i u^{(i)}$ with a $(\pm\lfloor h/2 \rfloor)$-sum of vectors from S_i. After the substitution, we obtain a $(\pm\lfloor h/2 \rfloor$-zero-sum. To see it is nontrivial, we recall that some c_i is nonzero and the corresponding $u^{(i)}$ is also nonzero, which means $c_i u^{(i)}$ is nonzero and hence its substitution cannot be a trivial sum.

Finally we remark that the runtime involves $m+1$ executions of $\mathcal{A}$ with time T each, and m executions of Lemma 3.3 with time $\mathrm{poly}(m, \log h)$ each. □

3.1 Simple SIS^∞ algorithms

An immediate starting point for Corollary 3.4 is the simple algorithm to find a nontrivial linear dependence in $n+1$ vectors. This leads to the following theorem for the SIS^∞ problem.

Theorem 3.5. *Let $q \geq 3$ be a prime and $1 \leq k \leq \lfloor q/2 \rfloor$ be an integer power of 2. Then there is a deterministic $\mathrm{poly}(m, \log q)$ time algorithm, given $m = (n+1)^k$ input vectors in $\mathbb{F}_q^n$, that outputs a nontrivial $(\pm\lfloor q/(2k) \rfloor)$-zero-sum.*

Proof. The base case $k = 1$ is the simple algorithm to find a nontrivial $(\pm\lfloor q/2 \rfloor)$-zero-sum in $n+1$ vectors, which runs in time $\mathrm{poly}(n, \log q)$. The inductive case $k \geq 2$ follows from Corollary 3.4 with the inductive hypothesis for $k/2$, and noticing $\lfloor \lfloor q/k \rfloor / 2 \rfloor = \lfloor q/(2k) \rfloor$. □

To improve Theorem 3.5, we explore the same ideas of dimension reduction and sparser zero-sum in Section 2 to get a better starting point. We begin with the idea of finding sparser zero-sums.

Lemma 3.6. *Let $r \geq 1$ be an arbitrary integer. Given $\ell + r$ vectors in a linear subspace of $\mathbb{F}_q^n$ with dimension ℓ, we can find in deterministic $\mathrm{poly}(n, r, \log q)$ time a nontrivial $(\pm\lfloor q/2 \rfloor)$-zero-sum that uses at most*

$$\frac{1 - q^{-1}}{1 - q^{-r}} \cdot \ell + r$$

vectors.

Proof. Similar proof as Lemma 2.3 where we set $q = 3$ and $r = \lceil \log_3(\ell+1) \rceil$. Let $\ell' \leq \ell$ be the rank of the input vectors. By an invertible linear transform and in $\mathrm{poly}(n, r)$ time, we can assume that we are given standard basis $e_1, \dots, e_{\ell'}$ and $v_1, \dots, v_r \in \mathrm{span}\{e_1, \dots, e_{\ell'}\}$ as r extra vectors. Then we consider linear combination $v_\beta = \beta_1 v_1 + \cdots + \beta_r v_r$ where the coefficient vector $\beta = (\beta_1, \dots, \beta_r)$ is uniform in $\mathbb{F}_q^r \setminus \{0^r\}$. A direct calculation shows, in expectation, the number of nonzero entries in v_β is at most $\frac{1-q^{-1}}{1-q^{-r}} \cdot \ell'$. Therefore, once we can find such a v_β, these nonzero entries can be canceled using $e_1, \dots, e_{\ell'}$. The total number of used vectors is

$$\underbrace{r}_{\text{for } v_\beta} + \underbrace{\frac{1 - q^{-1}}{1 - q^{-r}} \cdot \ell'}_{\text{for } e_1, \dots, e_{\ell'}} \leq \frac{1 - q^{-1}}{1 - q^{-r}} \cdot \ell + r.$$

Finally we remark that searching for such a v_β can be done efficiently in time $\mathrm{poly}(n, r)$ using the standard derandomization of conditional expectation to sequentially fix $\beta_1, \dots, \beta_r$.[8] □

[8] Since the vectors are in $\mathbb{F}_q^n$, for each $\beta_i \in \mathbb{F}_q$ there are at most $\min\{q, n+1\}$ choices with different conditional expectation. Hence the enumeration at each derandomization step is $\min\{q, n+1\} = O(n)$, independent of q.

We note that the $r = 1$ case of Lemma 3.6 is the most basic algorithm to find nontrivial linear dependence in $\ell + 1$ vectors, as used in Theorem 3.5. We now incorporate the idea of dimension reduction.

Theorem 3.7. *Let $q > 3$ be a prime and let $r \geq 1$ be an arbitrary integer. Given*

$$m \geq \frac{1 - q^{-1}}{1 - q^{-r}} \cdot \frac{n(n+1)}{2} + r(n+1)$$

vectors in $\mathbb{F}_q^n$, we can find in deterministic $\mathrm{poly}(m, \log q)$ time a nontrivial $(\pm\lfloor q/4 \rfloor)$-zero-sum.

Proof. The proof mimics the proof of Theorem 1.2. Define

$$m_\ell = \frac{1 - q^{-1}}{1 - q^{-r}} \cdot \frac{\ell(\ell+1)}{2} + r(\ell+1).$$

We prove a more general statement: given $m \geq m_\ell$ vectors in a linear subspace V of $\mathbb{F}_q^n$ with dimension ℓ, we can efficiently find a $(\pm\lfloor q/4 \rfloor)$-zero-sum. Theorem 3.7 follows by setting $\ell = n$.

The base case $\ell = 0$ is trivial, as $m_0 = r \geq 1$ and the given vector must be 0. For the induction step $\ell \geq 1$, since $m_\ell \geq \ell + r$, we apply Lemma 3.6 on the first $\ell + r$ vectors. We will find a set T of size at most $r + \frac{1-q^{-1}}{1-q^{-r}} \cdot \ell$ and a nontrivial $(\pm\lfloor q/2 \rfloor)$-zero-sum from T. Then by Lemma 3.3, this is turned into a $(\pm\lfloor q/2 \rfloor \to \pm\lfloor q/4 \rfloor)$-reducible vector u from T.

Now if $u = 0$ then we are done, since u itself is a nontrivial (± 1)-sum. Otherwise write $V = \mathrm{span}\{u\} \oplus W$, where W has dimension $\ell - 1$. We will put T aside and work on the remaining $m' = m - |T| \geq m_{\ell-1}$ vectors. Each remaining vector v_i can be written as $c_i u + w_i$ for some $c_i \in \mathbb{F}_q$ and $w_i \in W$. Applying induction to the $m' \geq m_{\ell-1}$ many w_i's, we obtain a nontrivial $(\pm\lfloor q/4 \rfloor)$-zero-sum of the w_i's. Now the associated collection of original remaining vectors v_i's sums to $c \cdot u$ for some $c \in \mathbb{F}_q$. Since u is reducible, this can be replaced by a $(\pm\lfloor q/4 \rfloor)$-sum from T, after which we obtain a nontrivial $(\pm\lfloor q/4 \rfloor)$-zero-sum as desired. □

Remark 3.8. A more optimized parameter choice for Theorem 3.7 is to pick a potentially different r_i for each dimension i and define

$$m_\ell = \sum_{i=0}^{\ell} \left(\frac{1 - q^{-1}}{1 - q^{-r_i}} \cdot i + r_i \right),$$

where each $r_i \geq 1$ is an integer and it automatically guarantees that $m_\ell \geq \ell + r_\ell$. By setting $r_0 = 1$ and $r_i = \lceil \log_q(i+1) \rceil$ for $i \geq 1$, we have

$$m_\ell \leq 1 + \sum_{i=1}^{\ell} \left(\left(1 - q^{-1}\right)(i+1) + \lceil \log_q(i+1) \rceil \right) = \frac{1 - q^{-1}}{2} \cdot \ell^2 + \ell \log_q(\ell+1) + O(\ell).$$

However this does not affect the asymptotics in Theorem 1.6.

With Theorem 3.7, we improve Theorem 3.5 and Theorem 1.6 follows immediately.

Theorem 3.9. *Let $q \geq 5$ be a prime and $2 \leq k \leq \lfloor q/2 \rfloor$ be an integer power of 2. Let $r \geq 1$ be an arbitrary integer. Given*

$$m \geq \left(\frac{1 - q^{-1}}{1 - q^{-r}} \cdot \frac{n(n+1)}{2} + r(n+1) \right)^{k/2}$$

vectors in $\mathbb{F}_q^n$, we can find in deterministic $\mathrm{poly}(m, \log q)$ time a nontrivial $(\pm\lfloor q/(2k) \rfloor)$-zero-sum.

Proof. The base case $k = 2$ is the algorithm in Theorem 3.7. The inductive case $k \geq 4$ follows from Corollary 3.4 with the inductive hypothesis for $k/2$, and noticing $\lfloor \lfloor q/k \rfloor /2 \rfloor = \lfloor q/(2k) \rfloor$. □

Different choice of r leads to a different bound in Theorem 3.9. For $r = 1$, it generalizes the bound in Theorem 2.2. For $r = \lceil \log_q(n+1) \rceil$, it generalizes the bound in Theorem 1.2. These choices of r readily prove Theorem 1.6.

We also remark that Theorem 3.9 (and even its weaker version Theorem 3.5) improves the result of [II24] by a factor of $q^{O(q \log q)}$.

3.2 $\mathbb{F}_q^n$-Subset-Sum on random inputs

Now we turn to Subset-Sum, a particularly interesting case of the more general CIS problem. Imran and Ivanyos [II24] proved a reduction from (± 1)-zero-sum algorithms to subset-zero-sum algorithms that works for worst-case instances but suffers from a blowup of $O(\log q)$ on the exponent. We identify a more efficient reduction for average-case instances.

We will take a detour to $(0, 1, 2)$-zero-sum, which means the zero-sum only uses coefficients in $0, 1, 2$. For consistency, we use $(0, 1)$-zero-sum to denote subset-zero-sum. Again, we say they are nontrivial if not all coefficients are 0.

With the help of a worst-case (± 1)-zero-sum algorithm, we show how to combine $(0, 1, 2)$-zero-sums to obtain a $(0, 1)$-zero-sum.

Lemma 3.10. *Let $q \geq 3$ be a prime. Suppose there is a deterministic algorithm $\mathcal{A}$, given m input vectors in $\mathbb{F}_q^n$ and running in time T, that outputs a nontrivial (± 1)-zero-sum.*

Then given vectors $v_1, \ldots, v_{m'} \in \mathbb{F}_q^n$ and assuming they form m disjoint nontrivial $(0, 1, 2)$-zero-sums, i.e., for each $i \in [m]$, we have

$$\sum_{j \in S_i} \alpha_j v_j = 0 \quad \text{where } S_1, \ldots, S_m \subseteq [m'] \text{ are disjoint and nonempty and each } \alpha_j \in \{1, 2\}, \tag{10}$$

we can find in time $T + \mathrm{poly}(m', n)$ a nontrivial $(0, 1)$-zero-sum in $v_1, \ldots, v_{m'}$.

Proof. If for some $i \in [m]$ we have $\alpha_j = 1$ for all $j \in S_i$, then $\sum_{j \in S_i} v_j = 0$ and we are done. Similarly if for some $i \in [m]$ we have $\alpha_j = 2$ for all $j \in S_i$, then $\sum_{j \in S_i} v_j = \sum_{j \in S_i} 2 \cdot v_j = 0$ and we are also done.

Now assume for every $i \in [m]$, there is some $j, j' \in S_i$ satisfying $\alpha_j = 1$ and $\alpha_{j'} = 2$. For every $i \in [m]$, rearrange Equation (10) as

$$u_i + 2 \cdot u_i' = 0 \quad \text{where } u_i = \sum_{j \in S_i \colon \alpha_j = 1} v_j \text{ and } u_i' = \sum_{j \in S_i \colon \alpha_j = 2} v_j. \tag{11}$$

By our assumption, each u_i, u_i' is nontrivial. Now we run $\mathcal{A}$ on $u_1', \ldots, u_m'$ and obtain a nontrivial (± 1)-zero-sum

$$\sum_{i \in S} u_i' - \sum_{i \in T} u_i' = 0,$$

where $S, T \subseteq [m]$ are disjoint and not both empty. With Equation (11), we obtain the following zero-sum

$$\sum_{i \in S} u_i' + \sum_{i \in T} u_i = \left(\sum_{i \in S} u_i' - \sum_{i \in T} u_i' \right) + \sum_{i \in T} (u_i + 2 \cdot u_i') = 0,$$

which is a desired nontrivial $(0, 1)$-zero-sum by expanding each u_i, u_i'. The runtime is obvious from the procedure above. □

A worst-case (± 1)-zero-sum algorithm is provided in Section 3. To apply Equation (10), it remains to obtain $(0,1,2)$-zero-sums. Intuitively $0,1,2$ is just a additive shift of $-1,0,1$, which suggests to use (± 1)-zero-sum algorithms to find $(0,1,2)$-zero-sums. This is formalized as the following lemma.

Lemma 3.11. *Let $q \geq 3$ be a prime. Suppose there is a deterministic algorithm $\mathcal{A}$, given m input vectors in $\mathbb{F}_q^n$ and running in time T, that outputs a nontrivial (± 1)-zero-sum.*

Let $\epsilon \in (0,1]$ be arbitrary and define $d = \lceil \log(q/\epsilon) \rceil$. Then there is a deterministic algorithm $\mathcal{B}$, given $dm+1$ input vectors in $\mathbb{F}_q^n$ and running in time $dT + \mathrm{poly}(m, q, \log(1/\epsilon))$, that outputs a nontrivial $(0,1,2)$-zero-sum with probability at least $1-\epsilon$ over uniform random inputs.

Proof. Let $v_1, \ldots, v_{dm+1}$ be the input vectors and define $v^* = -\sum_{i \in [dm+1]} v_i$. If $v^* = 0$, then we are done, as $\sum_{i \in [dm+1]} v_i = -v^* = 0$ is a trivial $(0,1,2)$-zero-sum. Now we assume $v^* \neq 0$. If we can find a (± 1)-sum of $v_1, \ldots, v_{dm}$ that equals v^*, then by expanding v^* and rearranging, we obtain a $(0,1,2)$-zero-sum. Note that such a $(0,1,2)$-zero-sum is nontrivial since v_{dm+1} always has coefficient 1.

To find the target (± 1)-sum, we write $\mathbb{F}_q^n = \mathrm{span}\{v^*\} \oplus W$ where W has dimension $n-1$. Let $i^* \in [n]$ be a nonzero coordinate of v^*, i.e., $v^*[i^*] \neq 0$. For each $i \in [dm]$, we express v_i as $c_i v^* + w_i$ where $c_i = v_i[i^*]/v^*[i^*] \in \mathbb{F}_q$ and $w_i = v_i - c_i v^* \in W$. Since $v_1, \ldots, v_{dm+1}$ are independent and uniform, we know $c_1, \ldots, c_{dm}$ are independent and uniform even when we condition on v^*.

Now we divide $v_1, \ldots, v_{dm}$ into d batches $S_1, \ldots, S_d$ of m vectors. On each batch $j \in [d]$, we run $\mathcal{A}$ on $w_i, i \in S_j$ to obtain a nontrivial (± 1)-zero-sum in time T:

$$\sum_{i \in S_j} \alpha_i w_i = 0 \quad \text{where } \alpha_i \in \{0, \pm 1\} \text{ are not all zero.}$$

This means $\sum_{i \in S_j} \alpha_i v_i = c_j v^*$ where $\beta_j = \sum_{i \in S_j} \alpha_i c_i$. Suppose there is some set $T \subseteq [d]$ that $\sum_{j \in T} \beta_j = 1$. We can use the textbook dynamic programming algorithm to find such a set in time $\mathrm{poly}(q, d)$, then obtain a desired (± 1)-sum

$$\sum_{j \in T} \sum_{i \in S_j} \alpha_i w_i = v^*.$$

It remains to show such a T exists with probability $1-\epsilon$. Since $\alpha_i, i \in S_j$ are not all zero and the c_i's are independent and uniform, $\beta_1, \ldots, \beta_d \in \mathbb{F}_q$ are independent and uniform. For each $\emptyset \neq T \subseteq [d]$, define $X_T \in \{0,1\}$ as the indicator that $\sum_{j \in T} \beta_j = 1$. Let $X = \sum_{\emptyset \neq T \subseteq [d]} X_T$. Then we have $\mathbf{E}[X] = (2^d - 1)/q$ and $\mathbf{Var}[X] = (2^d - 1)(q^{-1} - q^{-2})$, since $\mathbf{E}[X_T] = 1/q$ and the X_T's are pairwise independent. In addition, since

$$\mathbf{Var}[X] = \sum_{k \geq 0} \mathbf{Pr}[X = k] \cdot (k - \mathbf{E}[X])^2 \geq \mathbf{Pr}[X = 0] \cdot \mathbf{E}[X]^2,$$

we have

$$\mathbf{Pr}[\text{such a } T \text{ does not exist}] = \mathbf{Pr}[X = 0] \leq \frac{\mathbf{Var}[X]}{\mathbf{E}[X]^2} = \frac{q-1}{2^d - 1} \leq \frac{q}{2^d} \leq \epsilon.$$

This completes the proof. □

At this point, we are ready to present our algorithm that finds a subset-zero-sum in random inputs.

Theorem 3.12. *Let $q \geq 5$ be a prime and define k the unique integer power of 2 with $q/4 < k \leq \lfloor q/2 \rfloor$. Let $\epsilon \in (0,1]$ be arbitrary and let $1 \leq r \leq n$ be an arbitrary integer. Given*

$$m \geq \left(\frac{1-q^{-1}}{1-q^{-r}} \cdot \frac{n(n+1)}{2} + r(n+1)\right)^k \cdot O\left(\log(q)\log(1/\epsilon)\right)$$

uniform random vectors in $\mathbb{F}_q^n$, with probability at least $1-\epsilon$ we can find in deterministic $\mathrm{poly}(m,q)$ time a nontrivial $(0,1)$-zero-sum.

Proof. For any real number $x \geq 1$, there is a unique integer power of 2 that is greater than x and at most $\lfloor 2x \rfloor$. This shows the existence and uniqueness of our choice of k, which also guarantees $\lfloor q/(2k) \rfloor = 1$. Let

$$\overline{m} = \left(\frac{1-q^{-1}}{1-q^{-r}} \cdot \frac{n(n+1)}{2} + r(n+1)\right)^{k/2}$$

be the bound in Theorem 3.9 such that we can always efficiently find a nontrivial (± 1)-zero-sum in $\overline{m}$ input vectors.

Let $d = O(\log q)$. Then by Lemma 3.11, with probability 0.99, we can find a nontrivial $(0,1,2)$-zero-sum in $d \cdot \overline{m} + 1$ uniform random input vectors. Hence by standard concentration, with probability at least $1-\epsilon$ we can find $\overline{m}$ disjoint nontrivial $(0,1,2)$-zero-sums if we run the above procedure on $O(\overline{m}\log(1/\epsilon))$ many independent batches of $d \cdot \overline{m} + 1$ uniform random input vectors. Finally we apply Lemma 3.10 on these $(0,1,2)$-zero-sums to obtain a nontrivial $(0,1)$-zero-sum.

The total number of input vectors is $O(\overline{m}\log(1/\epsilon)) \cdot (d \cdot \overline{m} + 1)$ as claimed. □

The reduction in [II24] produces a bound of $m \geq q^{O(q\log^2 q)} n^{O(q\log q)}$, whereas our Theorem 3.12 produces $m \geq n^{O(q)}$.

3.3 Simple CIS algorithms: large primes

Finally we use the simple tools developed so far to give simple algorithms for the CIS problem and prove Theorem 1.12. This will already dequantize, and in fact greatly improve, Theorem 1.11, when the field size q grows.

Recall that the CIS problem will only allow coefficients from some allowed set $A \subseteq \mathbb{F}_q$. For consistency with previous sections, we use A-sum to denote a vector sum whose coefficients are all from A. We say it is A-zero-sum if the vector sum also equals 0. Then a solution to the CIS problem is equivalent to a nontrivial A-zero-sum.

For any $a \in \mathbb{F}_q$, we use aA to denote the set $\{aa' : a' \in A\}$ and use $A + a$ to denote the set $\{a + a' : a' \in A\}$.

The following reduction is a natural generalization of Lemma 3.11. In particular, Lemma 3.11 is equivalent to setting $A = \{0, \pm 1\}, a = 1, b = 1$ in Lemma 3.13.

Lemma 3.13. *Let $q \geq 3$ be a prime and let $A \subseteq \mathbb{F}_q$ be nonempty. Suppose there is a deterministic algorithm $\mathcal{A}$, given m input vectors in $\mathbb{F}_q^n$ and running in time T, that outputs a nontrivial A-zero-sum.*

Let $\epsilon \in (0,1]$ be arbitrary and define $d = \lceil \log(q/\epsilon) \rceil$. Fix arbitrary $a \in \mathbb{F}_q \setminus \{0\}, b \in \mathbb{F}_q$ and define $B = aA + b$. Then there is a deterministic algorithm $\mathcal{B}$, given $dm+1$ input vectors in $\mathbb{F}_q^n$ and running in time $dT + \mathrm{poly}(m, q, \log(1/\epsilon))$, that outputs a nontrivial B-zero-sum with probability at least $1-\epsilon$ over uniform random inputs.

Proof. We safely assume $a = 1$ since any nontrivial zero-sum with coefficients in $A + (b/a)$ can be dilated, by multiplying every coefficient with a, as a nontrivial zero-sum with coefficients in $a(A + (b/a)) = B$. If $b = 0$, then we just run $\mathcal{A}$ on m input vectors to obtain a nontrivial A-zero-sum. Now we assume $a = 1$ and $b \neq 0$.

Let $v_1, \dots, v_{dm+1}$ be input vectors. We define $v^* = -b \sum_{i \in [dm+1]} v_i$. It suffices to find an A-sum of $v_1, \dots, v_{dm}$ that equals v^*, then by expanding v^* and rearranging, we obtain a B-zero-sum. Note that such a B-zero-sum is nontrivial since v_{dm+1} has coefficient $b \neq 0$. The rest of the analysis is identical to the proof of Lemma 3.11. □

Combining Lemma 3.13 with the SIS$^\infty$ algorithms in Section 3, we can handle any allowed coefficient set with long arithmetic progression.

Definition 3.14 (Arithmetic Progression (AP))**.** Let $A \subseteq \mathbb{F}_q$ have size $c \geq 2$. We say A is a t-AP (equivalently, an AP of length t) if there are $x \in \mathbb{F}_q$ and $0 \neq y \in \mathbb{F}_q$ such that $A = \{x, x + y, x + 2y, \dots, x + (t - 1)y\}$. We say A contains a t-AP if some $A' \subseteq A$ is a t-AP.

Corollary 3.15. *Let $q \geq 5$ be a prime and let $A \subseteq \mathbb{F}_q$ be nonempty. Assume A contains an AP of length $1 + 2\lfloor q/(2k) \rfloor$ where $k \geq 2$ that is an integer power of 2. Let $r \geq 1$ be an arbitrary integer. Given*

$$m \geq 1 + \left(\frac{1 - q^{-1}}{1 - q^{-r}} \cdot \frac{n(n+1)}{2} + r(n+1) \right)^{k/2} \cdot \lceil \log(q/\epsilon) \rceil$$

uniform random vectors in $\mathbb{F}_q^n$, with probability at least $1 - \epsilon$ we can find in deterministic $\mathrm{poly}(m, q)$ time a nontrivial A-zero-sum.

Proof. Without loss of generality assume A is the $(2k + 1)$-AP. By Lemma 3.13, it suffices to have a worst-case algorithm for nontrivial $(\pm\lfloor q/(2k) \rfloor)$-zero-sums, for which we use Theorem 3.9. □

In the case of large q, we have known results in arithmetic combinatorics that guarantee long APs. The following fact is due to Lev [Lev00] and, in Section A, we provide a short proof for completeness.

Fact 3.16 ([Lev00])**.** Let $q \geq 5$ be a prime. Let $A \subseteq \mathbb{F}_q$ be an arbitrary set of size $|A| \geq q - \log_4(q + 2)$. Then A contains an AP of length $(q + 1)/2$.

Putting together Fact (3.16) and Corollary 3.15, we obtain our simple CIS algorithms for large primes.

Theorem 3.17. *Let $q \geq 5$ be a prime and let $A \subseteq \mathbb{F}_q$ be an arbitrary set of size $|A| \geq q - \log_4(q+2)$. Let $r \geq 1$ be an arbitrary integer. Given*

$$m \geq 1 + \left(\frac{1 - q^{-1}}{1 - q^{-r}} \cdot \frac{n(n+1)}{2} + r(n+1) \right) \cdot \lceil \log(q/\epsilon) \rceil$$

uniform random vectors in $\mathbb{F}_q^n$, with probability at least $1 - \epsilon$ we can find in deterministic $\mathrm{poly}(m, q)$ time a nontrivial A-zero-sum.

Proof. By Fact (3.16), A contains an AP of length $(q + 1)/2$. Since $(q + 1)/2 \geq 1 + 2\lfloor q/4 \rfloor$, we set $k = 2$ and apply Corollary 3.15. □

Finally we prove Theorem 1.12, which is a simple deduction from Theorem 3.17.

Proof of Theorem 1.12. Since $q > 4^{k-1}$ and $k \geq 2$, we know $q \geq 5$ and the number of allowed coefficients is $q - k + 1 \geq q - \log_4(q + 2)$. Hence we set $r = 1$ and $\epsilon \in (0, 1]$ to be a small constant then apply Theorem 3.17. This proves Theorem 1.12. □

4 Beyond the halving trick

In this section, we expand our toolbox by developing more general reductions.

To give some motivation, consider the setting where we want to find a nontrivial $(\pm\lfloor q/6\rfloor)$-zero-sum. So far we only have the halving trick, and thus we will pay a *quartic* blowup by halving twice, which ends up providing a nontrivial $(\pm\lfloor q/8\rfloor)$-zero-sum and is more than we need. Our new reductions in this section will allow us to pay a *cubic* cost to find nontrivial $(\pm\lfloor q/6\rfloor)$-zero-sums.

Notation. We will generalize existing notions as follows. For $A, B \subseteq \mathbb{F}_q$, we define $A - B = \{a - b \colon a \in A, b \in B\}$. We use $\pm A$ to denote $A \cup (-A)$. We say sets $A_1, \dots, A_k$ is a nonempty disjoint partition of A if $A_1, \dots, A_k$ are all nonempty and pairwise disjoint and $A = A_1 \cup \cdots \cup A_k$.

Definition 4.1 (General Vector Sum)**.** Let $v_1, \dots, v_m \in \mathbb{F}_q^n$ be vectors and let $H \subseteq \mathbb{F}_q$ be nonempty. We say vector u is an H-sum of the v_i's if we can choose $\alpha_i \in H$ for each $i \in [m]$ such that $u = \sum_{i\in[m]} \alpha_i v_i$; and it is an H-zero-sum if $u = 0$.

We say u is s-sparse if it has at most s nonzero coefficients. Let $\emptyset \neq A \subseteq \mathbb{F}_q \setminus \{0\}$. We say u is A-nontrivial if for any $a \in A$ there exists some $i \in [m]$ such that $\alpha_i \in \{a, -a\}$.

In comparison with Definition 3.1, a $(\pm h)$-sum is equivalent to an H-sum where $H = \{-h, -h+1, \dots, h\}$ and a nontrivial sum is equivalent to an A-nontrivial sum for some $\emptyset \neq A \subseteq \mathbb{F}_q \setminus \{0\}$. In particular, a $(\pm\lfloor q/2\rfloor)$-zero-sum is an $\mathbb{F}_q$-zero-sum.

Definition 4.2 (General Reducible Vector)**.** Let $v_1, \dots, v_m \in \mathbb{F}_q^n$ be vectors and let $H, H' \subseteq \mathbb{F}_q$ and let $\emptyset \neq A \subseteq \mathbb{F}_q \setminus \{0\}$. We say that vector u is $(H \to H')$-reducible for the v_i's if

- u is a nontrivial (± 1)-sum of the v_i's (i.e., u is a $\{1\}$-nontrivial $\{0, \pm 1\}$-sum of the v_i's);
- for every $c \in H$, the vector $c \cdot u$ is an H'-sum of the v_i's.

We say u is s-sparse if there exists a set $S \subseteq [m]$ of size $|S| \leq s$ such that u is $(H \to H')$-reducible for $v_i, i \in S$. Let $\emptyset \neq A \subseteq \mathbb{F}_q \setminus \{0\}$. We say u is A-nontrivial if for every $c \in H$, $c \cdot u$ is an A-nontrivial H'-sum.

In comparison with Definition 3.2, an $(\pm h \to \pm h')$-reducible vector is a $(H \to H')$-reducible vector where $H = \{-h, -h+1, \dots, h\}$ and $H' = \{-h', -h'+1, \dots, h'\}$.

Whenever we construct a reducible vector u, we will provide a procedure to support efficient query: given $c \in H$, we can express $c \cdot u$ as an H'-sum (or A-nontrivial H'-sum) of the v_i's in deterministic time $\mathrm{poly}(m, n, \log q)$. Hence to ease presentation, we will omit this detail and only refer to the construction of reducible vectors.

4.1 Basic zero-sum algorithms for specific nontriviality

Given Definition 4.1, it is natural to ask for basic algorithms that ensure the specific A-nontriviality. It turns out that it is not too hard to modify the existing Lemma 3.6 to obtain what we need.

Lemma 4.3. *Let $\emptyset \neq A \subseteq \mathbb{F}_q \setminus \{0\}$ and let $r \geq 1$ be an arbitrary integer. Given $n + r|A|$ vectors in $\mathbb{F}_q^n$, we can find in deterministic $\mathrm{poly}(n, r, |A|, \log q)$ time an s-sparse A-nontrivial $\mathbb{F}_q$-zero-sum, where*

$$s = \frac{1 - q^{-1}}{1 - q^{-r}} \cdot n + r|A|.$$

Proof. Denote $A = \{a_1, \dots, a_s\}$ where $s = |A|$. Let $\ell \le n$ be the rank of the input vectors. By an invertible linear transform and in $\mathrm{poly}(n, r, |A|)$ time, we can assume that we are given standard basis $e_1, \dots, e_\ell$ and $v_1^{(i)}, \dots, v_s^{(i)}, i \in [r]$ as rs extra vectors that lie in the span of $e_1, \dots, e_\ell$.

For each $i \in [r]$ we compute

$$u^{(i)} = a_1 v_1^{(i)} + \cdots + a_s v_s^{(i)}. \tag{12}$$

Then by the same proof as Lemma 3.6, we can find $\beta_1, \dots, \beta_r \in \mathbb{F}_q$ not all zero such that

$$u_\beta = \beta_1 u^{(1)} + \cdots + \beta_r u^{(r)} \tag{13}$$

has at most $\frac{1-q^{-1}}{1-q^{-r}} \cdot \ell$ nonzero entries.

Since $\beta_1, \dots, \beta_r$ are not all zero, we can safely assume $\beta_j = 1$ for some $j \in [r]$. Hence by substituting Equation (12) into Equation (13), we know that $a_1, \dots, a_s$ all appear as coefficients of u_β, which implies that u_β is A-nontrivial.

Finally we use the standard basis $e_1, \dots, e_\ell$ to cancel the nonzero entries of u_β, which provides the desired zero-sum. □

4.2 From zero-sum to reducible vector

Similar to Lemma 3.3, we can obtain $(H \to H')$-reducible vectors from H-zero-sums. However things are more complicated in the general setting.

We start with the simplest case that we partition H into two parts and one of which becomes (part of) H'. It is encouraged to keep in mind the example of Lemma 3.3 where $H = \{-h, \dots, h\}$ and $H' = \{-\lfloor h/2 \rfloor, \dots, \lfloor h/2 \rfloor\}$.

Lemma 4.4. *Let $H \subseteq \mathbb{F}_q$ and let $\emptyset \ne A \subseteq \mathbb{F}_q \setminus \{0\}$. Suppose there is a deterministic algorithm $\mathcal{A}$, given m vectors in $\mathbb{F}_q^n$ and running in time T, that outputs an s-sparse A-nontrivial H-zero-sum.*

Let H_0, H_1 be a nonempty disjoint partition of H. Assume $A_0 := (\pm A) \cap H_0$ and $A_1 := (\pm A) \cap H_1$ are nonempty. Let $B_1 \subseteq \mathbb{F}_q$ be arbitrary such that $H_1 \subseteq \pm B_1$. Define

$$H' = (\pm H_0) \cup (B_1 - B_1).$$

Then there is a deterministic algorithm $\mathcal{B}$, given m vectors in $\mathbb{F}_q^n$ and running in time $T + \mathrm{poly}(m, n, \log q)$, that outputs an s-sparse A_0-nontrivial $(H_1 \to H')$-reducible vector.

Proof. Let $v_1, \dots, v_m \in \mathbb{F}_q^n$ be the input vectors. We apply $\mathcal{A}$ to obtain an s-sparse A-nontrivial H-zero-sum $\alpha_1 v_1 + \cdots + \alpha_m v_m$. Let $S \subseteq [m]$ of size $|S| \le s$ be the nonzero coefficients. Compute $S_0 = \{j \in S \colon a_j \in H_0\}$ and $S_1 = \{j \in S \colon a_j \in H_1\}$, which is a disjoint partition of S. Moreover, since it is A-nontrivial and $A_0 \subseteq H_0, A_1 \subseteq H_1$ are nonempty, we know S_0, S_1 are nonempty.

For each $j \in S_1$, we define $\beta_j \in \{\pm 1\}$ such that $\alpha_j \beta_j \in B_1$, i.e., $\beta_j = 1$ if $\alpha_j \in B_1$; and $\beta_j = -1$ if $-\alpha_1 \in B_1$. Now we can rewrite the A-nontrivial B-zero-sum as

$$\sum_{j \in S_0} \alpha_j v_j + \sum_{j \in S_1} \alpha_j \beta_j \cdot \beta_j v_j = 0. \tag{14}$$

By our construction above, $\sum_{j \in S_0} \alpha_j v_j$ is an A_0-nontrivial H_0-sum and $\sum_{j \in S_1} \alpha_j \beta_j \cdot \beta_j v_j = \sum_{j \in S_1} \alpha_j v_j$ is an A_1-nontrivial H_1-sum.

At this point, we define

$$u = \sum_{j \in S_1} \beta_j v_j. \tag{15}$$

We will prove that u is a A_0-nontrivial $(H_1 \to H')$-reducible vector.

As mentioned above, $\sum_{j\in S_1} \alpha_j\beta_j \cdot \beta_j v_j$ is an A_1-nontrivial H_1-sum; hence u is a nontrivial (± 1)-sum of $v_j, j \in S$. Let $c \in H_1$ and, by symmetry, assume $c \in B_1$. Then we have

$$\begin{aligned} c \cdot u &= \sum_{j\in S_1} c\beta_j v_j - \left(\sum_{j\in S_0} \alpha_j v_j + \sum_{j\in S_1} \alpha_j\beta_j \cdot \beta_j v_j \right) && \text{(by Equation (14))} \\ &= \underbrace{-\sum_{j\in S_0} \alpha_j v_j}_{P} + \underbrace{\sum_{j\in S_1} (c - \alpha_j\beta_j) \cdot \beta_j v_j}_{Q}. && \text{(by Equation (15))} \end{aligned}$$

Note P has coefficients in $\pm H_0 \subseteq H'$ and is A_0-nontrivial; and Q has coefficients in $B_1 - B_1 \subseteq H'$. Hence $u = P + Q$ is an A-nontrivial $(H \to H')$-reducible vector. Since we only use vector in S, it is also s-sparse. The runtime follows directly from the description above. □

In the following Lemma 4.5, we proceed to the general case, where we partition H into more $k + 1 \geq 3$ parts. Analogously, it is helpful to keep in mind the example where $H = \{-h, \dots, h\}$, $H_0 = \{-\lfloor h/(k+1)\rfloor, \dots, \lfloor h/(k+1)\rfloor\}$, and each $H_i, i \in [k]$ is an interval of length at most $\lceil h/(k+1)\rceil$.

We will need the following standard notions on permutations. Let $k \geq 1$ be an integer. We use $\mathcal{S}_k \subsetneq [k]^k$ to denote the symmetric group on $[k]$ elements. Each $\pi = (\pi_1, \dots, \pi_k) \in \mathcal{S}_k$ defines the action $\pi(i) = \pi_i$ for all $i \in [k]$. We also define $\pi^{-1}(i) \in [k]$ to be the location of i in π, i.e., $\pi_{\pi^{-1}(i)} = i$. We use $\operatorname{sgn}(\pi) \in \{\pm 1\}$ to denote the sign of the permutation π.

Lemma 4.5. *Let $H \subseteq \mathbb{F}_q$ and let $\emptyset \neq A \subseteq \mathbb{F}_q \setminus \{0\}$. Suppose there is a deterministic algorithm $\mathcal{A}$, given $m(n)$ vectors in $\mathbb{F}_q^n$ and running in time $T(n)$, that outputs an $s(n)$-sparse A-nontrivial H-zero-sum.*

Let $H_0, H_1, \dots, H_k$ be a nonempty disjoint partition of H where $k \geq 2$. Assume $A_i := (\pm A) \cap H_i$ is nonempty for all $i = 0, \dots, k$. For $i \in [k]$, let $B_i \subseteq \mathbb{F}_q$ be arbitrary such that $H_i \subseteq \pm B_i$. Define

$$H' = (\pm H_0) \cup \bigcup_{i\in[k]} (B_i - B_i).$$

Define $m_1 = 1$ and, for $\ell = 2, 3, \dots, k+1$,

$$\begin{aligned} m_\ell &= (-1 + m_{\ell-1}) \cdot s(k^{k-\ell+1}n) + m(k^{k-\ell+1}n) \\ &= \prod_{j=1}^{\ell-1} s(k^{k-j}n) + \sum_{i=1}^{\ell-1} \Big(m(k^{k-i}n) - s(k^{k-i}n) \Big) \prod_{j=1}^{i-1} s(k^{k-j}n). \end{aligned}$$

Then there is a deterministic algorithm $\mathcal{B}$, given m_{k+1} vectors in $\mathbb{F}_q^n$ and running in time T', that outputs an s'-sparse A_0-nontrivial $(H_1 \cup \dots \cup H_k \to H')$-reducible vector, where

$$s' = \prod_{\ell=1}^{k} s(k^{k-\ell}n) \qquad \text{and} \qquad T' = \sum_{\ell=1}^{k} m_\ell \cdot T(k^{k-\ell}n) + \operatorname{poly}(m_{k+1}, n, k, \log q).$$

Proof. Let $v_1, \dots, v_{m_{k+1}}$ be the input vectors. We will build a forest of rooted trees of depth at most $k + 1$ by iteratively applying $\mathcal{A}$.

1. For each depth $\ell = 1, 2, \dots, k+1$, there are exactly m_ℓ nodes. The leaves are at depth $k+1$.
2. The jth leaf is simply denoted j. We define $Q^j = v_j \in \mathbb{F}_q^n$ and denote $Q^j[\emptyset] = v_j$.

3. For depth $\ell = k, k-1, \ldots, 1$, we first list all $m_{\ell+1}$ nodes, denoted $x_1, \ldots, x_{m_{\ell+1}}$, at depth $\ell+1$. Then we create m_ℓ nodes at depth ℓ as follows.

   Each time we create a depth-ℓ node z, we pick an arbitrary set $S \subseteq \{x_1, \ldots, x_{m_{\ell+1}}\}$ of the remaining depth-$(\ell+1)$ nodes of size $|S| = m(k^{k-\ell}n)$. Each $x \in S$ is already associated with some vector $Q^x \in \mathbb{F}_q^{k^{k-\ell}n}$; so we apply $\mathcal{A}$ on $Q^x, x \in S$ to obtain an $s(k^{k-\ell}n)$-sparse A-nontrivial H-zero-sum

$$\sum_{x \in T} \alpha_x Q^x = 0,$$

   where $T \subseteq S$ contains the nonzero coefficients and has size $|T| \le s(k^{k-\ell}n)$. Then we connect $x \in T$ as the child nodes of z and discard them from the remaining depth-$(\ell+1)$ nodes.

   Since each time we discard at most $s(k^{k-\ell}n)$ nodes, we are guaranteed with at least $m(k^{k-\ell}n)$ remaining ones for creating each one of the m_ℓ many z's. Runtime here is $m_\ell \cdot T(k^{k-\ell}n)$ for executions of $\mathcal{A}$.

   Finally we record the following quantities.

   (a) Nonzero coefficients α_x of the zero-sum above for each $x \in T$.

   (b) Nonempty sets $S_i^z \subseteq T$ for each $i = 0, \ldots, k$ corresponding to coefficients in H_i, such that $|S_0^z| + \cdots + |S_k^z| = |T| \le s(k^{k-\ell}n)$ and $S_i^z \cap S_{i'}^z = \emptyset$ whenever $i \ne i'$.
   The nonemptiness and disjointness are because the H-zero-sum is A-nontrivial, where $H_0, \ldots, H_k$ is a nonempty disjoint partition of H with nonempty $A_0 \subseteq H_0, \ldots, A_k \subseteq H_k$.

   (c) Signs $\beta_x \in \{\pm 1\}$ for each $x \in T$ such that $\alpha_x \beta_x \in B_i$ where $i = 0, 1, \ldots, k$ is the unique index that $x \in S_i^z$.

   (d) A_i-nontrivial H_i-sum $w_i^z = \sum_{x \in S_i^z} \alpha_x Q^x$ for each $i = 0, \ldots, k$, such that $w_0^z + \cdots + w_k^z = 0$.

   (e) Nontrivial (± 1)-sum $Q_i^z = \sum_{x \in S_i^z} \beta_x Q^x$ for each $i \in [k]$.
   Note that Q_i^z is nontrivial as $S_i^z \ne \emptyset$.

   (f) Vector $Q^z \in \mathbb{F}_q^{k^{k-\ell+1}n}$ defined by tupling $Q_1^z, \ldots, Q_k^z$.
   Alternatively, we view $Q^z \in (\mathbb{F}_q^n)^{k \times \cdots \times k}$ as a $(k-\ell+1)$-dimensional array. For $R = (r_1, \ldots, r_{k-\ell+1}) \in [k]^{k-\ell+1}$, we recursively define the Rth entry of Q^z, denoted $Q^z[R] \in \mathbb{F}_q^n$, as

$$Q^z[R] = Q_{r_1}^z[r_2, \ldots, r_{k-\ell+1}] = \sum_{x \in S_{r_1}^z} \beta_x Q^x[r_2, \ldots, r_{k-\ell+1}].$$

The total runtime T' consists of $\sum_{\ell=1}^k m_\ell \cdot T(k^{k-\ell}n)$ for executions of $\mathcal{A}$ and $\mathrm{poly}(m_{k+1}, n, k, \log q)$ for other processing time.

Note that there is a single node z_0 at depth 1 in the above construction. We use $\mathcal{T}$ to denote the depth-$(k+1)$ tree rooted at z_0. For our reducible vector u, we will only use input vectors that are leaves of $\mathcal{T}$. Since each depth-ℓ node has at most $s(k^{k-\ell}n)$ child nodes, $\mathcal{T}$ has at most $\prod_{\ell=1}^k s(k^{k-\ell}n)$ leaves. This proves the sparsity bound s'.

Now we show how to obtain a reducible vector u given $\mathcal{T}$. Recall that $\mathcal{S}_k \subsetneq [k]^k$ is the set of permutations on $[k]$. Define

$$\begin{aligned} u &= \sum_{\pi \in \mathcal{S}_k} \mathrm{sgn}(\pi) Q^{z_0}[\pi] \\ &= \sum_{\pi = (\pi_1, \ldots, \pi_k) \in \mathcal{S}_k} \mathrm{sgn}(\pi) \sum_{z_1 \in S_{\pi_1}^{z_0}} \beta_{z_1} \sum_{z_2 \in S_{\pi_2}^{z_1}} \beta_{z_2} \cdots \sum_{z_k \in S_{\pi_k}^{z_{k-1}}} \beta_{z_k} Q^{z_k}[\emptyset] \qquad \text{(by Item 3f)} \end{aligned}$$

$$= \sum_{\pi=(\pi_1,\dots,\pi_k)\in\mathcal{S}_k} \operatorname{sgn}(\pi) \sum_{z_1\in S^{z_0}_{\pi_1}} \beta_{z_1} \sum_{z_2\in S^{z_1}_{\pi_2}} \beta_{z_2} \cdots \sum_{z_k\in S^{z_{k-1}}_{\pi_k}} \beta_{z_k} v_{z_k} \qquad \text{(by Item 2)}$$

$$= \sum_{\pi=(\pi_1,\dots,\pi_k)\in\mathcal{S}_k} \operatorname{sgn}(\pi) \sum_{z_1\in S^{z_0}_{\pi_1},\dots,z_k\in S^{z_{k-1}}_{\pi_k}} \left(\prod_{\ell\in[k]} \beta_{z_\ell}\right) v_{z_k}. \qquad (16)$$

By Item 3b and Item 3e, u is a nontrivial (±1)-sum. To prove that u is an A_0-nontrivial $(H_1 \cup \cdots \cup H_k \to H')$-reducible vector, let $i^* \in [k]$ and $c \in H_{i^*}$ be arbitrary. By symmetry, we assume $c \in B_{i^*}$ and analyze $c \cdot u$.

Fix an arbitrary $\pi = (\pi_1, \dots, \pi_k) \in \mathcal{S}_k$. Let $j = \pi^{-1}(i^*)$. Define $\widetilde{\pi} = (\pi_1, \dots, \pi_{j-1})$ and $\overline{\pi} = (\pi_{j+1}, \dots, \pi_k)$; so $\pi = (\widetilde{\pi}, i^*, \overline{\pi})$. We use $\mathcal{T}[\widetilde{\pi}]$ to denote the set of nodes z can be reached by the following process: start with $z = z_0$; for each step $d = 1, \dots, j-1$, proceed to an arbitrary child node of z in $S^z_{\pi_d}$. We have the following observation.

Claim 4.6. *$\mathcal{T}[\widetilde{\pi}]$ is a nonempty set of depth-j nodes. Let $z_{j-1} \in \mathcal{T}[\widetilde{\pi}]$ be arbitrary. Then*

$$\sum_{i=0}^{k} \sum_{z_j\in S_i^{z_{j-1}}} \alpha_{z_j}\beta_{z_j} \sum_{z_{j+1}\in S^{z_j}_{\pi_{j+1}},\dots,z_k\in S^{z_{k-1}}_{\pi_k}} \left(\prod_{\ell=j}^{k} \beta_{z_\ell}\right) v_{z_k} = 0.$$

Proof. The nonemptiness and depth condition follow directly from Item 3b. By Item 3d, we have $\sum_{i=0}^{k} \sum_{z_j\in S_i^{z_{j-1}}} \alpha_{z_j} Q^{z_j} = w_0^{z_{j-1}} + \cdots + w_k^{z_{j-1}} = 0$. By Item 3f, each Q^{z_j} is a concatenation of $Q^{z_j}[R]$ for $R \in [k]^{k-j}$. So the above equation holds for every choice of R. Setting $R = \overline{\pi}$ and expanding with Item 3f, Item 2, and Item 3c proves the claim. □

Given Claim 4.6, we analyze $c \cdot u$ as follows:

$$c \cdot u = \sum_{\pi} \operatorname{sgn}(\pi) \sum_{z_1\in S^{z_0}_{\pi_1},\dots,z_k\in S^{z_{k-1}}_{\pi_k}} c \left(\prod_{\ell\in[k]} \beta_{z_\ell}\right) v_{z_k} \qquad \text{(by Equation (16))}$$

$$= \sum_{\pi} \operatorname{sgn}(\pi) \sum_{\substack{z_1\in S^{z_0}_{\pi_1},\dots,z_{j-1}\in S^{z_{j-2}}_{\pi_{j-1}} \\ j:=\pi^{-1}(i^*)}} \beta_{z_1}\cdots\beta_{z_{j-1}} \sum_{z_j\in S_{i^*}^{z_{j-1}}} c \sum_{z_{j+1}\in S^{z_j}_{\pi_{j+1}},\dots,z_k\in S^{z_{k-1}}_{\pi_k}} \left(\prod_{\ell=j}^{k} \beta_{z_\ell}\right) v_{z_k}$$

$$= \sum_{\pi} \operatorname{sgn}(\pi) \sum_{\substack{z_1\in S^{z_0}_{\pi_1},\dots,z_{j-1}\in S^{z_{j-2}}_{\pi_{j-1}} \\ j:=\pi^{-1}(i^*)}} \beta_{z_1}\cdots\beta_{z_{j-1}}$$

$$\cdot \left(\sum_{z_j\in S_{i^*}^{z_{j-1}}} c - \sum_{i=0}^{k} \sum_{z_j\in S_i^{z_{j-1}}} \alpha_{z_j}\beta_{z_j}\right) \sum_{z_{j+1}\in S^{z_j}_{\pi_{j+1}},\dots,z_k\in S^{z_{k-1}}_{\pi_k}} \left(\prod_{\ell=j}^{k} \beta_{z_\ell}\right) v_{z_k}. \qquad \text{(by Claim 4.6)}$$

By dividing the summation over i into different cases, we have $c \cdot u = L^* - L_0 - L_1$ where

$$L^* = \sum_{\pi} \operatorname{sgn}(\pi) \sum_{\substack{z_1\in S^{z_0}_{\pi_1},\dots,z_k\in S^{z_{k-1}}_{\pi_k} \\ j:=\pi^{-1}(i^*)}} (c - \alpha_{z_j}\beta_{z_j}) \left(\prod_{\ell\in[k]} \beta_{z_\ell}\right) v_{z_k}, \qquad \text{(via } i = i^*)$$

$$L_0 = \sum_{\pi} \operatorname{sgn}(\pi) \sum_{\substack{z_1 \in S^{z_0}_{\pi_1}, \dots, z_{j-1} \in S^{z_{j-2}}_{\pi_{j-1}} \\ j := \pi^{-1}(i^*) \\ z_j \in S_0^{z_{j-1}} \\ z_{j+1} \in S^{z_j}_{\pi_{j+1}}, \dots, z_k \in S^{z_{k-1}}_{\pi_k}}} \alpha_{z_j} \beta_{z_j} \left(\prod_{\ell \in [k]} \beta_{z_\ell} \right) v_{z_k}, \qquad \text{(via } i = 0\text{)}$$

and

$$L_1 = \sum_{i \in [k] \setminus \{0, i^*\}} \sum_{\pi} \operatorname{sgn}(\pi) \sum_{\substack{z_1 \in S^{z_0}_{\pi_1}, \dots, z_{j-1} \in S^{z_{j-2}}_{\pi_{j-1}} \\ j := \pi^{-1}(i^*) \\ z_j \in S_i^{z_{j-1}} \\ z_{j+1} \in S^{z_j}_{\pi_{j+1}}, \dots, z_k \in S^{z_{k-1}}_{\pi_k}}} \alpha_{z_j} \beta_{z_j} \left(\prod_{\ell \in [k]} \beta_{z_\ell} \right) v_{z_k}. \qquad \text{(via } i \notin \{0, i^*\}\text{)}$$

We use the following Claim 4.7 to show that $c \cdot u$ is an A_0-nontrivial H'-sum.

Claim 4.7. *L^*, L_0, L_1 are disjoint vector sums (i.e., they sum over disjoint sets of leaves of $\mathcal{T}$). Moreover, L^* is an H'-sum; L_0 is an A_0-nontrivial H'-sum; and L_1 is an H'-sum.*

Proof. We first simplify L_1. Let $\Lambda \subseteq [k]^k$ be the set of $\sigma \in [k]^k$ such that σ contains no i^* and exactly $k-1$ distinct elements, i.e., $\{\sigma_1, \dots, \sigma_k\} = [k] \setminus \{i^*\}$. Then there are distinct $j := j_\sigma \in [k]$ and $j' := j'_\sigma \in [k]$ such that $\sigma_j = \sigma_{j'}$. Define π_σ to be σ with σ_j replaced by i^*; and define π'_σ to be σ with $\sigma_{j'}$ replaced by i^* Then $\pi_\sigma, \pi'_\sigma \in \mathcal{S}_k$ and enumerating $\pi \in \mathcal{S}_k$ is equivalent as enumerating σ and going over π_σ, π'_σ. Therefore we can simplify L_1 as

$$L_1 = \sum_{\sigma \in \Lambda} \sum_{(j,\pi) \in \{(j_\sigma, \pi_\sigma), (j'_\sigma, \pi'_\sigma)\}} \operatorname{sgn}(\pi) \sum_{z_1 \in S^{z_0}_{\sigma_1}, \dots, z_k \in S^{z_{k-1}}_{\sigma_k}} \alpha_{z_j} \beta_{z_j} \left(\prod_{\ell \in [k]} \beta_{z_\ell} \right) v_{z_k} \qquad \text{(noticing } i = \sigma_j\text{)}$$

$$= \sum_{\sigma \in \Lambda} \sum_{z_1 \in S^{z_0}_{\sigma_1}, \dots, z_k \in S^{z_{k-1}}_{\sigma_k}} \operatorname{sgn}(\pi_\sigma) \left(\alpha_{z_{j_\sigma}} \beta_{z_{j_\sigma}} - \alpha_{z_{j'_\sigma}} \beta_{z_{j'_\sigma}} \right) \left(\prod_{\ell \in [k]} \beta_{z_\ell} \right) v_{z_k}, \tag{17}$$

where we used $\operatorname{sgn}(\pi_\sigma) = -\operatorname{sgn}(\pi'_\sigma)$ for the last equality.

We observe that L^* traverses $\mathcal{T}$ from root to leaf based on $\pi \in \mathcal{S}_k \subsetneq [k]^k$; L_0 traverses based on $\pi \in \mathcal{S}_k$ with i^* replaced by 0; and L_1, via Equation (17), traverses based on $\sigma \in \Lambda$. Since the above criterion does not overlap, they are disjoint vector sums.

Finally we analyze the coefficients.

- For L^*, we recall that $c \in B_{i^*}$ and, by Item 3c, $\alpha_{z_j} \beta_{z_j} \in B_{i^*}$, which shows that L^* is a $B_{i^*} - B_{i^*} \subseteq H'$-sum.
- For L_0, we notice that $\alpha_{z_j} \in H_0$. Given other signs multiplied together, L_0 is a $(\pm H_0) \subseteq H'$-sum. Note that L_0 is A_0-nontrivial by Items 3b and 3d.
- For L_1 and any fixed $\sigma \in \Lambda$, let $i = \sigma_{j_\sigma} \in [k] \setminus \{i^*\}$. Then by Item 3c, we know both $\alpha_{z_{j_\sigma}} \beta_{z_{j_\sigma}}$ and $\alpha_{z_{j_\sigma}} \beta_{z_{j'_\sigma}}$ are in B_i. Hence L_1 is a $\bigcup_{i \in [k] \setminus \{i^*\}} (B_i - B_i) \subseteq H'$-sum. □

Finally we remark that, given c, we can efficiently express $c \cdot u$ as the target A_0-nontrivial H'-sum by computing L^*, L_0 directly with their definition and computing L_1 with its simplified form Equation (17). □

Remark 4.8. We note that in Item 3 when depth $\ell = 1$, we do not have to use the H-zero-sum algorithm with specific A-nontriviality. Instead, it suffices to use any nontrivial H-zero-sum algorithm. This will still guarantee the nontriviality of u in Equation (16). Since this saving does not affect the final bound much, we do not introduce further complications here.

4.3 From reducible vector to zero-sum

Similar to Corollary 3.4, we can improve zero-sum algorithm from reducible vectors.

Lemma 4.9. *Let $H \subseteq \mathbb{F}_q$ and let $\emptyset \neq A \subseteq \mathbb{F}_q \setminus \{0\}$. Suppose there is a deterministic algorithm $\mathcal{A}$, given $m(n)$ vectors in $\mathbb{F}_q^n$ and running in time $T(n)$, that outputs an $s(n)$-sparse A-nontrivial H-zero-sum.*

Let $H_0, H_1, \dots, H_k$ be a nonempty disjoint partition of H where $k \geq 1$. Assume $A_i := (\pm A) \cap H_i$ is nonempty for all $i = 0, \dots, k$. For $i \in [k]$, let $B_i \subseteq \mathbb{F}_q$ be arbitrary such that $H_i \subseteq \pm B_i$. Define

$$H' = (\pm H_0) \cup \bigcup_{i \in [k]} (B_i - B_i).$$

Define $m_1 = 1$ and, for $\ell = 2, 3, \dots, k+1$,

$$m_\ell = \prod_{j=1}^{\ell-1} s(k^{k-j} n) + \sum_{i=1}^{\ell-1} \Big(m(k^{k-i} n) - s(k^{k-i} n) \Big) \prod_{j=1}^{i-1} s(k^{k-j} n).$$

Then there is a deterministic algorithm $\mathcal{A}'$, given $m(n) \cdot m_{k+1}$ vectors in $\mathbb{F}_q^n$ and running in time $T(n) + m(n) \cdot T'$, that outputs an $(s(n) \cdot s')$-sparse A_0-nontrivial H'-zero-sum, where

$$s' = \prod_{\ell=1}^{k} s(k^{k-\ell} n) \qquad \text{and} \qquad T' = \sum_{\ell=1}^{k} m_\ell \cdot T(k^{k-\ell} n) + \mathrm{poly}(m_{k+1}, n, k, \log q).$$

Proof. We apply Lemma 4.4 or Lemma 4.5 with $\mathcal{A}$ to obtain an algorithm $\mathcal{B}$ that constructs an s'-sparse A_0-nontrivial $(H_1 \cup \dots \cup H_k \to H')$-reducible vector for m_{k+1} input vectors. We run $\mathcal{B}$ on $m = m(n)$ disjoint batches of input vectors to produce reducible vectors $u^{(1)}, \dots, u^{(m)}$. Then we apply $\mathcal{A}$ on $u^{(1)}, \dots, u^{(m)}$ to obtain an A-nontrivial H-zero-sum $\beta_1 u^{(1)} + \dots + \beta_m u^{(m)} = 0$. For each $j \in [m]$, if $\beta_j \in H_i$ for some $i \in [k]$, then we replace $\beta_j u^{(j)}$ with an A_0-nontrivial H'-sum by the reducibility of $u^{(j)}$. Crucially since $\beta_1 u^{(1)} + \dots + \beta_m u^{(m)} = 0$ is A-nontrivial and $\pm A$ hits some (in fact, every) $H_i, i \in [k]$, the above substitution always happens, which means the final H'-zero-sum is A_0-nontrivial.

The total number of vectors used is $m(n)$ batches of m_{k+1} vectors; the runtime is mainly $m(n)$ executions of $\mathcal{B}$ and one execution of $\mathcal{A}$; and the sparsity is $s(n) \cdot s'$ where $s(n)$ is the sparsity of the β_j's and s' is the sparsity of each nonzero $\beta_j u^{(j)}$ (and its substitution). □

Remark 4.10. If we do not need to enforce the specific A_0-nontriviality in Lemma 4.9, then we do not have to use the A-nontrivial H-zero-sum algorithm to combine the A_0-nontrivial $(H_1 \cup \dots \cup H_k \to H')$-reducible vectors. This still ensures a nontrivial H'-zero-sum in the end. However the saving does not affect the final bound much.

4.4 Improved SIS^∞ algorithms

The new reduction allows us to improve the bounds for the SIS^∞ problem and prove Theorem 1.5.

The previous halving trick corresponds to the case of $k = 1$ and we will need to iterate many times to get solutions with smaller weights. Now with the ability to handle general k, we do not have to iterate and can finish in one shot.

The following Theorem 4.11 removes the assumption of k being an integer power of 2 in Theorem 3.9 and proves Theorem 1.5 immediately.

Theorem 4.11. *Let $q \geq 5$ be a prime and $2 \leq k \leq \lfloor q/2 \rfloor$ be an integer. Given*

$$(k-1)^{(k-1)(k-2)/2}(n+k)^k$$

vectors in $\mathbb{F}_q^n$, we can find in deterministic $\mathrm{poly}(m, k, \log q)$ time a nontrivial $(\pm\lfloor q/(2k) \rfloor)$-zero-sum.

Proof. Define $B_0, \ldots, B_{k-1} \subsetneq \{1, \ldots, \lfloor q/2 \rfloor\}$ be contiguous intervals of length at most $\lfloor q/(2k) \rfloor + 1$, where $B_0 = \{1, \ldots, \lfloor q/(2k) \rfloor\}$. Let $H_0 = (\pm B_0) \cup \{0\}$ and $H_i = \pm B_i$ for $1 \leq i \leq k-1$. Since $H_0 = \{-\lfloor q/(2k) \rfloor, \ldots, \lfloor q/(2k) \rfloor\}$, it is equivalent to find a nontrivial H_0-zero-sum.

Observe that $H_0, \ldots, H_{k-1}$ form a nonempty disjoint partition of $H := \mathbb{F}_q$. Let $A' \subseteq \mathbb{F}_q$ be of size k such that A' contains exactly one element of H_i for each $0 \leq i \leq k-1$. By Lemma 4.3 with $r = 1$, given $m(n)$ vectors in $\mathbb{F}_q^n$, we can efficiently find an $s(n)$-sparse A'-nontrivial $\mathbb{F}_q$-zero-sum where $m(n) = s(n) = n + k$. Let this be the algorithm $\mathcal{A}$ for Lemma 4.9.

We apply Lemma 4.9 with $H, H_0, \ldots, H_{k-1}$ and $B_1, \ldots, B_{k-1}$. Since each $B_i - B_i \subseteq H_0$, we have $H' = H_0$. Hence we can efficiently compute a nontrivial H_0-zero-sum given $\overline{m}$ input vectors, where

$$\begin{aligned}\overline{m} &= (n+k) \cdot \prod_{j=1}^{k-1} \left((k-1)^{k-1-j} n + k \right) \\ &\leq (n+k)^k \prod_{j=1}^{k-1} (k-1)^{k-1-j} = (k-1)^{(k-1)(k-2)/2}(n+k)^k.\end{aligned}$$

□

4.5 Improved $\mathbb{F}_q^n$-Subset-Sum algorithms

Recall that in Section 3.2 we use worst-case SIS^∞ algorithms to derive average-case Subset-Sum algorithms. Since we now have better SIS^∞ algorithms, it is natural to expect a similar improvement for Subset-Sum.

This is formalized as the following Theorem 4.12, which should be compared with Theorem 3.12. Note that Theorem 1.9 follows immediately from Theorem 4.12.

Theorem 4.12. *Let $q \geq 5$ be a prime and define $k = \lfloor (q+3)/4 \rfloor$. Let $\epsilon \in (0, 1]$ be arbitrary. Given*

$$m \geq (k-1)^{(k-1)(k-2)}(n+k)^{2k} \cdot O(\log(q)\log(1/\epsilon)) = q^{O(q^2)} \cdot n^{2k} \cdot \log(1/\epsilon)$$

uniform random vectors in $\mathbb{F}_q^n$, with probability at least $1 - \epsilon$ we can find in deterministic $\mathrm{poly}(m)$ time a nontrivial $(0, 1)$-zero-sum.

Proof. Note that our choice of k satisfies $k > q/4$ and thus $\lfloor q/(2k) \rfloor = 1$. Then the analysis is identical to the proof of Theorem 3.12, except that we use Theorem 4.11 as the (± 1)-zero-sum algorithm. □

It will also be useful to generalize $\{0,1\}$ to handle all sets of size 2. This can be done with another application of Lemma 3.13.

Theorem 4.13. *Let $q \geq 5$ be a prime and define $k = \lfloor (q+3)/4 \rfloor$. Let $\epsilon \in (0,1]$ be arbitrary and let $A \subseteq \mathbb{F}_q$ be of size 2. Given*

$$m \geq q^{O(q^2)} \cdot n^{2k} \cdot \log^2(1/\epsilon)$$

uniform random vectors in $\mathbb{F}_q^n$, with probability at least $1-\epsilon$ we can find in deterministic $\mathrm{poly}(m)$ time a nontrivial A-zero-sum.

Proof. Observe that $A = a\{0,1\} + b$ for some $a \in \mathbb{F}_q \setminus \{0\}$ and $b \in \mathbb{F}_q$. Let $\mathcal{A}$ be the deterministic $(0,1)$-zero-sum algorithm in Theorem 4.12 and we apply Lemma 3.13 to obtain nontrivial A-zero-sum from nontrivial $(0,1)$-zero-sums.

While Lemma 3.13 is stated for deterministic worst-case algorithm $\mathcal{A}$, it is also easy to see that it works for deterministic average-case algorithm $\mathcal{A}$, which is what we have here. The only change will be to ensure the success of individual runs of $\mathcal{A}$. This leads to the following set-up: we pick $d = \lceil \log(2q/\epsilon) \rceil$ in Lemma 3.13 and run d independent $\mathcal{A}$'s. We provide $\overline{m} = q^{O(q^2)} \cdot n^{2k} \cdot \log(2d/\epsilon)$ vectors for each individual $\mathcal{A}$ to produce a nontrivial $(0,1)$-zero-sum. By our parameter choice, each $\mathcal{A}$ succeeds with probability $1 - \epsilon/(2d)$; and, upon the success of all $\mathcal{A}$'s executions, the final conversion to A-zero-sum succeeds with probability $1-\epsilon/2$. By a union bound, the overall success probability is $1-\epsilon$ over the uniform random inputs while our algorithm is deterministic. The total number of input vectors is $1 + d \cdot \overline{m} = q^{O(q^2)} \cdot n^{2k} \cdot \log^2(1/\epsilon)$ as claimed. □

4.6 Improved CIS algorithms

Now we are ready to improve the simple CIS algorithm in Section 3.3 and prove Theorem 1.13.

Recall that in the CIS problem, we are given an arbitrary set of allowed coefficients. To handle such generality, in Lemma 4.5 we will set $B_1, \ldots, B_k$ as singleton sets to ensure $B_i - B_i = \{0\}$ for all $i \in [k]$. This is formalized in the following Theorem 4.14.

Theorem 4.14. *Let $q \geq 5$ be a prime. Let $1 \leq a_1 < \cdots < a_k \leq \lfloor q/2 \rfloor$ be arbitrary where $1 \leq k < \lfloor q/2 \rfloor$. Define $\overline{A} = \{\pm a_1, \ldots, \pm a_k\}$ and $A = \mathbb{F}_q \setminus \overline{A}$. Then given*

$$m \geq k^{k(k-1)/2}(n+k+1)^{k+1}$$

vectors in $\mathbb{F}_q^n$, we can find in deterministic $\mathrm{poly}(m, \log q)$ time a nontrivial A-zero-sum.

Proof. Let $a_0 \in A \setminus \{0\}$ be arbitrary. Define $A' = \{a_0, a_1, \ldots, a_k\}$. By Lemma 4.3 with $r = 1$, given $m(n)$ vectors in $\mathbb{F}_q^n$, we can efficiently find an $s(n)$-sparse A'-nontrivial $\mathbb{F}_q$-zero-sum where $m(n) = s(n) = n + k + 1$. Let this be the algorithm $\mathcal{A}$ for Lemma 4.9.

Define $H = \mathbb{F}_q$, $H_0 = A$, and $H_i = \{\pm a_i\}, B_i = \{a_i\}$ for $i \in [k]$. Then apply Lemma 4.9, where $H' = A$. Therefore we can efficiently compute a nontrivial A-zero-sum given $\overline{m}$ input vectors, where

$$\begin{aligned}\overline{m} &= (n+k+1) \cdot \prod_{j=1}^{k} \left(k^{k-j} n + k + 1\right) \\ &\leq (n+k+1)^{k+1} \cdot \prod_{j=1}^{k} k^{k-j} = k^{k(k-1)/2}(n+k+1)^{k+1}.\end{aligned}$$

□

Using Lemma 3.13, we can shift and dilate A in Theorem 4.14 to handle more general cases.

Theorem 4.15. *Let $q \geq 5$ be a prime. Let $1 \leq a_1 < \cdots < a_k \leq \lfloor q/2 \rfloor$ be arbitrary where $1 \leq k < \lfloor q/2 \rfloor$. Define $\overline{A} = \{\pm a_1, \ldots, \pm a_k\}$ and $A = \mathbb{F}_q \setminus \overline{A}$.*

Fix arbitrary $a \in \mathbb{F}_q \setminus \{0\}, b \in \mathbb{F}_q$ and define $B = aA + b$. Let $\epsilon \in (0, 1]$ be arbitrary. Then given

$$m \geq 1 + k^{k(k-1)/2}(n + k + 1)^{k+1} \cdot \lceil \log(q/\epsilon) \rceil$$

uniform random vectors in $\mathbb{F}_q^n$, with probability at least $1 - \epsilon$ we can find in deterministic $\mathrm{poly}(m, q)$ time a nontrivial B-zero-sum.

Proof. Follows immediately by combining Theorem 4.14 and Lemma 3.13. □

Observe that in Theorem 4.15 we pay a cost of roughly n^{k+1} to remove $2k$ elements, though these elements need to be paired up. This already hints an improvement over Theorem 1.11, as they pay a cost of roughly n^{2k+1} to remove only $2k$ elements. In general, whenever we have two paired elements that need to be discarded, we only incur a linear cost, in contrast to the quadratic cost in [CLZ22].

By Theorem 4.15, we look into the following arithmetic combinatorial structure: up to some additive shift, we want to find some $-x, x \in \overline{A}$ while guaranteeing some $-y, 0, y \in A$ to ensure $k < \lfloor q/2 \rfloor$. This leads to the following result, the proof of which is deferred to Section A.

Fact 4.16. Let $q \geq 3$ be a prime. Let $A \subseteq \mathbb{F}_q$ and define $c = |\mathbb{F}_q \setminus A|$. There exist $x, y, z \in \mathbb{F}_q$ such that the following holds.

1. If $2 \leq c < q$, then $x \neq 0$, $z \in A$, and $z - x, z + x \notin A$.
2. In addition to the conditions in Item 1, if $c < (q + 1)/2$, then $y \neq 0$ and $z - y, z + y \in A$.

The following fact completes the edge case in Item 2 of Fact (4.16). Its proof is also presented in Section A. A similar result over the integers is due to Erdős and Turán [ET36].

Fact 4.17. Let $q \geq 11$ be a prime. Let $A \subseteq \mathbb{F}_q$ and define $c = |\mathbb{F}_q \setminus A|$. Assume $c = (q + 1)/2$. Then A contains a 3-AP.

The above results fall into the richer literature of quantitative Szemerédi's theorem. Using more advanced techniques, the bound in Fact (4.17) can be strengthened to $c \leq (1 - o(1)) \cdot q$ and 3-AP can be extended into longer APs. Numerous works are along this line and we refer interested readers to [KM23, LSS24] for recent breakthroughs.

We emphasize that we choose to work with the simpler but worse bounds because (1) they are sufficient for our purposes, (2) they admit simpler proofs that we can present for self-containedness, and (3) they hold for small primes with explicit constants.

Now we use Fact (4.16) and Fact (4.17), together with algorithms in Section 4.4 and Section 4.5, to prove the following Theorem 4.18, from which Theorem 1.13 follows immediately.

Theorem 4.18. *Let $q \geq 3$ be a prime. Let $1 \leq c \leq q-2$ be an integer and let $B \subseteq \mathbb{F}_q$ be arbitrary of size $q - c$. Let $\epsilon \in (0, 1]$ be arbitrary. Then given m uniform random vectors in $\mathbb{F}_q^n$, with probability at least $1 - \epsilon$ we can find in deterministic $\mathrm{poly}(m, q)$ time a nontrivial B-zero-sum, where*

- *$m \geq n^{c+1} \cdot O(\log(q/\epsilon) \log(1/\epsilon))$ if $c = 1$ or ($c = 3$ and $q = 5$).*
- *$m \geq n^c \cdot c^{O(c^2)} \cdot \log(q/\epsilon) \log(1/\epsilon)$ if otherwise, i.e., ($c \geq 2$ and $q > 5$) or ($c = 2$ and $q = 5$).*

Proof. Let $\overline{A} \subseteq \mathbb{F}_q$, which has size $1 \leq c \leq q - 2$. Then we have the following cases.

The $q = 3$ case. Note that $c = 1$ and $|B| = 2$, which is the (shifted) $\mathbb{F}_3^n$-Subset-Sum problem. Hence we use Theorem 1.2 and Lemma 3.13 to handle this case. The number of input vectors is

$$O(n^2) \cdot \lceil \log(3/\epsilon) \rceil = n^{c+1} \cdot O(\log(1/\epsilon)).$$

The $c = 1$ and $q \geq 5$ case. Define $A = \{-\lfloor q/2 \rfloor + 1, \ldots, \lfloor q/2 \rfloor - 1\}$. Under an additive shift $b \in \mathbb{F}_q$, $A + b$ is contained in B. Hence we can apply Theorem 4.15 with $k = 1$ to obtain a nontrivial $A + b \subseteq B$-zero-sum. The number of input vectors is

$$1 + (n+2)^2 \lceil \log(q/\epsilon) \rceil = n^{c+1} \cdot O\left(\log(q/\epsilon)\right).$$

The $2 \leq c \leq (q-1)/2$ case. By Fact (4.16) we find $z \in \mathbb{F}_q$ and $x, y \in \mathbb{F}_q \setminus \{0\}$ such that $z - x, z + x \notin B$ and $z - y, z, z + y \in B$. Define $A = B - z$ and $\overline{A} = \mathbb{F}_q \setminus A$. Then $-x, x \in \overline{A}$ and $-y, 0, y \in A$. This means that we can find $1 \leq a_1 < \cdots < a_k \leq \lfloor q/2 \rfloor$ such that $\overline{A} \subseteq \{\pm a_1, \ldots, \pm a_k\}$; in particular, $1 \leq k \leq c - 1 < \lfloor q/2 \rfloor$ since $\pm x \in \overline{A}$. Hence we can apply Theorem 4.15 with k to obtain a nontrivial $A + b \subseteq B$-zero-sum. The number of input vectors is

$$1 + k^{k(k+1)/2}(n+2)^{k+1} \cdot \lceil \log(q/\epsilon) \rceil \leq c^{O(c^2)} \cdot n^c \cdot \log(q/\epsilon).$$

The $(q+1)/2 \leq c \leq q-2$ and $q \equiv 3 \pmod 4$ case. In this case, we pick an arbitrary $A \subseteq B$ of size 2 and apply Theorem 4.13 to obtain a nontrivial $A \subseteq B$-zero-sum. The number of input vectors is

$$q^{O(q^2)} \cdot n^{2\lfloor (q+3)/4 \rfloor} \cdot \log^2(1/\epsilon) \leq c^{O(c^2)} \cdot n^c \cdot \log^2(1/\epsilon),$$

where we compute $2\lfloor (q+3)/4 \rfloor = (q+1)/2 \leq c$.

The $(q+3)/2 \leq c \leq q-2$ and $q \equiv 1 \pmod 4$ case. Here we still reduce B into a set of size 2 and obtain a same bound as above. The number of input vectors is

$$q^{O(q^2)} \cdot n^{2\lfloor (q+3)/4 \rfloor} \cdot \log^2(1/\epsilon) \leq c^{O(c^2)} \cdot n^c \cdot \log^2(1/\epsilon),$$

where we compute $2\lfloor (q+3)/4 \rfloor = (q+3)/2 \leq c$.

The $c = (q+1)/2$ and $q \geq 11$ case. Let $A = \{0, \pm 1\}$. By Fact (4.17), we can find a 3-AP in B. Hence for some $a \in \mathbb{F}_q \setminus \{0\}$ and $b \in \mathbb{F}_q$, we have $aA + b \subseteq B$. By applying Theorem 4.11 with $k = \lfloor (q+3)/4 \rfloor$, we have $\lfloor q/(2k) \rfloor = 1$ and thus we can efficiently find a nontrivial A-zero-sum given $q^{O(q^2)} n^{\lfloor (q+3)/4 \rfloor}$ vectors. Then by Lemma 3.13, we can find nontrivial $aA + b \subseteq B$-zero-sums with an extra blowup of $O(\log(q/\epsilon))$. Hence the number of input vectors is

$$q^{O(q^2)} n^{\lfloor (q+3)/4 \rfloor} \cdot O(\log(q/\epsilon)) \leq c^{O(c^2)} \cdot n^c \cdot \log(1/\epsilon),$$

where we compute $\lfloor (q+3)/4 \rfloor \leq (q+1)/2 = c$.

The $c = (q+1)/2$ and $q = 5$ case. Now $c = 3$ and $|B| = q - c = 2$. By Theorem 4.13, the number of input vectors is

$$q^{O(q^2)} \cdot n^{2\lfloor (q+3)/4 \rfloor} \cdot \log^2(1/\epsilon) = n^{c+1} \cdot O\left(\log^2(1/\epsilon)\right),$$

where we compute $2\lfloor (q+3)/4 \rfloor = 4 = c + 1$. □

4.7 Further optimizations

Here we discuss extra tricks that can improve the sample complexity m of the results in this section.

Exploring sparsity. For clean presentation, we apply Lemma 4.9 with the simplest choice $r = 1$ throughout the section. However it is easy to see that a better choice would be $r \approx \log_q(n)$ when q is relatively small compared to n. Then the bound in Lemma 4.3 would be $s \approx (1 - q^{-1})n$, which becomes a multiplicative saving of roughly $(1-q^{-1})^k$ for m in Lemma 4.9 and all later applications.

Dimension reduction. Since we apply Lemma 4.9 directly with $H = \mathbb{F}_q$ through the section, the dimension reduction idea from Theorem 2.2 applies. This can save another multiplicative factor of $1/k$ in Theorem 4.11, Theorem 4.12, and Theorem 4.13; and a factor of $1/(k+1)$ in Theorem 4.14 and Theorem 4.15.

In a bit more detail, if $H = \mathbb{F}_q$, then in Lemma 4.9 each time we construct a reducible vector, we can safely project later vectors into its complementary space. This in general saves a factor of $1/(\ell+1)$ as $\sum_{i \le n} i^\ell \approx n^{\ell+1}/(\ell+1)$, where each i^ℓ is roughly the cost of Lemma 4.9 with $\ell + 1$ partitioned sets for input vectors in a subspace of dimension i.

Iterative application. All our results in this section use only a single shot of Lemma 4.9. It is natural to wonder if some iterative application similar to Theorem 3.9 will be beneficial. Consider the SIS$^\infty$ problem where we want to find a nontrivial $(\pm\lfloor q/(2k)\rfloor)$-zero-sum. If we apply Lemma 4.9 once, then we obtain Theorem 4.11 and a bound of $m \approx k^{k^2/2} n^k$ samples. Now we show how to improve it when k can be factored as a product of small integers.

Say $k = q_1 \cdots q_t$ where each $q_i \ge 2$ is an integer. Assume the field size q is much smaller than the vector dimension n for simplicity. Define $k_i = q_1 \cdots q_i$ where $k_0 = 1$ and $k_t = k$. We will iteratively apply Lemma 4.9 to $(\pm\lfloor q/(2k_i)\rfloor)$-zero-sum algorithms from $(\pm\lfloor q/(2k_{i-1})\rfloor)$-zero-sum algorithms. While there is some nontriviality subtlety that needs to be preserved in order to repeatedly apply Lemma 4.9, it will just be a minor factor due to $q \ll n$.

The starting point is the $(\pm\lfloor q/2\rfloor)$-zero-sum algorithm $\mathcal{A}_0$ from Lemma 4.3 that uses roughly $m_0(n) = n$ input vectors. Then we apply Lemma 4.9 to obtain a $(\pm\lfloor q/(2k_1)\rfloor)$-zero-sum algorithm that uses roughly $m_1(n) = p_1^{p_1^2/2} n^{p_1}$ input vectors. Then we apply Lemma 4.9 to divide q_2 which ends up needing roughly

$$m_2(n) \approx \prod_{j=1}^{q_2} m_1\left(q_2^{q_2-j} n\right) \approx \prod_{j=1}^{q_2} \left(q_2^{q_2-j} \cdot q_1^{q_1^2/2} n^{q_1}\right) \approx q_1^{q_1^2 q_2/2} \cdot q_2^{q_2^2/2} \cdot n^{q_1 q_2}$$

input vectors. Similar calculation shows that in the end, we can find $(\pm\lfloor q/(2k)\rfloor)$-zero-sum when the number of input vectors is

$$m_t(n) \approx n^{q_1 \cdots q_t} \cdot \prod_{i=1}^{t} q_i^{(q_i^2/2)\cdot\prod_{j=i+1}^{t} q_j}.$$

If each q_i is roughly the same around $r := k^{1/t}$, then it simplifies to

$$m_t(n) \approx n^k \cdot \prod_{i=1}^{t} r^{r^{t-i+2}/2} = n^k \cdot \left(r^{1/2}\right)^{\sum_{i=1}^{t} r^{t+2-i}} \approx n^k \cdot \left(r^{1/2}\right)^{r^{t+1}} = n^k \cdot r^{kr/2}.$$

Therefore, if r is small (say, constant), it improves the $k^{k^2/2}$ prefactor significantly to $O(1)^k$; and the worst case is k being prime and $t = 1$, for which it falls back to the one-shot case unless we are willing to sacrifice dependence on n.

5 Discussion

In this section, we describe improvements, generalizations, and open directions; and also discuss prior works related to our paper.

5.1 On our results

Targeted sum and more general CIS. While our work focuses on the zero-sum case in both the worst-case and average-case setting, it is meaningful to consider the case where we want to target some vector other than 0. This is a "closest-vector problem (CVP)" analogue, thinking of SIS$^\infty$ as a "shortest-vector problem (SVP)". The targeted version is at least as hard as the zero-sum version, although one might have an intuition it is "not too much" harder — at least in the average-case. There *are* some differences though; for example, whereas $\mathbb{F}_q^n$-Subset-Sum is guaranteed to have a solution whenever $m > (q-1)n$ (see Fact (5.2)), this is not true in the targeted case. For the average case where input vectors are uniformly at random, our algorithm works with minor change.

In addition, one may want to study the generalization of CIS where the ith coefficient has its own allowed set $A_i \subseteq \mathbb{F}_q$. Since our arithmetic combinatorial results depend only on the size of the allowed sets, they also work here; same for the reduction to handle translation and dilation (Lemma 3.13). Therefore our results on the CIS problem hold with the same bound in this more general setting.

Better runtime. We can make the runtime of our algorithms more explicit. Recall that our algorithms rely heavily on the weight reduction, either through the halving trick in Section 3 or through the more general trick in Section 4. It is easy to see that the reduction itself is computationally efficient in linear time, and the main runtime bottleneck lies in the base case where we find a nontrivial (sparse) linear dependence with every field element as an allowed coefficient. Such base case algorithms are presented in Lemma 2.3, Lemma 3.6, and Lemma 4.3; and they boil down to the following simple linear algebra problems: find basis vectors in input vectors and make some invertible linear transform.

Such problems naturally benefit from fast matrix multiplication algorithms. Let $2 \le \omega < 2.372$ denote a constant such that $n \times n$ matrix multiplication (and inversion) over $\mathbb{F}_q$ can be done in deterministic classical time $n^\omega \operatorname{polylog}(n, q)$. See [ADW+25] and references within for latest results. The following Fact (5.1) provides what we need, and its proof can be found in Section B.

Fact 5.1. There is a deterministic algorithm, given any $m \ge n$ vectors in $\mathbb{F}_q^n$ and running in time $n^{\omega-1}m \cdot \operatorname{polylog}(n, q)$, that outputs a maximal linearly independent set among the vectors.

Using Fact (5.1), the runtime of our SIS$^\infty$ algorithms can be optimized to $n^{\omega-1}m \cdot \operatorname{polylog}(n, q)$, and the runtime of our Subset-Sum and CIS algorithms is $\left(n^{\omega-1}m + \operatorname{poly}(q)\right) \cdot \operatorname{polylog}(n, q)$. By way of comparison, the CLZ quantum algorithm takes time at least m^ω; so, for example, our $\mathbb{F}_3^n$-Subset-Sum algorithm takes $n^{\omega+1}$ classical time, whereas CLZ's is $n^{2\omega}$ quantum time.

Better sample complexity. The sample complexity m is the main object we optimize in the paper. In Section 4, we prioritized clarity over optimality and obtained simple bounds that have targeted dependence on n but worse dependence in terms of other parameter like q (field size) or k (the partition number in Lemma 4.9). In Section 4.7, we discussed tricks that bring in additional savings on existing bounds in various settings. However it remains a challenge to significantly improve the dependence on q and k for all cases; and it is even less clear how to (or whether it is possible to) improve the dependence on n.

For concreteness, we ask if the $k^{O(k^2)}n^k$ bound in Theorem 4.11 can be improved to $O(n^k)$ (which would match Theorem 3.9 but hold for general k) or even $n^{o(k)}$.

Also for the $\mathbb{F}_q^n$-Subset-Sum problem, we ask if it is possible to solve worst-case instances with $O_q\left(n^{O(q)}\right)$ input vectors, which, if true, would improve the $O_q\left(n^{O(q\log q)}\right)$ bound in [II24] and match the average-case complexity. Finally for the $\mathbb{F}_3^n$-Subset-Sum problem, we ask if it is easy to solve with only $m = o(n^2)$ input vectors.

General finite field. While all the finite fields in our paper are prime field, most of our results work for general finite fields. However it may not be necessary to do so as we can always view the field elements as a vector of number in a prime field. In a bit more detail, let p be a prime and $t \geq 1$ be an integer such that $q = p^t$. We can view $\mathbb{F}_q$ as a vector space $\mathbb{F}_p^t$; thus dimension-n input vectors $v_1, \ldots, v_m$ over field $\mathbb{F}_q$ can be converted into dimension-tn input vectors $v'_1, \ldots, v'_m$ over field $\mathbb{F}_p$. In addition, any zero-sum of $v'_1, \ldots, v'_m$ is a zero-sum of $v_1, \ldots, v_m$ with the same set of coefficients. Hence we may alternatively just use the zero-sum algorithms over $\mathbb{F}_p$.

Ring of integers modulo q. Another meaningful generalization is to consider input vectors with entries in $\mathbb{Z}/q\mathbb{Z}$ where $q \geq 2$ is an integer. This reduces to $\mathbb{F}_q$ if q is a prime. For composite number q, our algorithms are still applicable with minor tweak. Take results in Section 4 for example. The basic zero-sum algorithm in Section 4.1 is fine as we can do Gaussian elimination with the subtractive Euclidean algorithm. Then the reductions in Section 4.2 and Section 4.3 are already good as only addition and subtraction are involved.

On the other hand, tweaking our algorithms may not be necessary. There is a simple reduction, which is essentially [CLZ22, Appendix A] and is credited to Regev, that converts a $(\pm h_1)$-zero-sum algorithm $\mathcal{A}_1$ over $\mathbb{Z}/q_1\mathbb{Z}$ and a $(\pm h_2)$-zero-sum algorithm $\mathcal{A}_2$ over $\mathbb{Z}/q_2\mathbb{Z}$ into a $(\pm h_1h_2)$-zero-sum algorithm over $\mathbb{Z}/(q_1q_2\mathbb{Z})$. The observation is that $v \equiv 0 \pmod{q_1q_2}$ is equivalent to (1) $v \equiv 0 \pmod{q_1}$ and (2) $(v/q_1) \equiv 0 \pmod{q_2}$. Therefore we can first run $\mathcal{A}_1$ to obtain zero-sums modulo q_1 (i.e., satisfying condition (1)); then run $\mathcal{A}_2$ on those zero-sums divided by q_1 to obtain zero-sums modulo q_2 (i.e., satisfying condition (2)). The final sample complexity is the sample complexities multiplied.

5.2 Works by Imran, Ivanyos, Sanselme, and Santha

The most comparable papers to the present one are a line of work by Ivanyos and coauthors [ISS07, IS17, II24]. These papers are all motivated by the Chevalley–Warning theorem and by designing *quantum* algorithms for versions of the Hidden Subgroup Problem; yet, all contain intermediate results that can be viewed as classical algorithms for SIS$^\infty$ or CIS.

The work of Ivanyos–Sanselme–Santha [ISS07] considered the problem of finding nontrivial solutions to (worst-case) systems of "diagonal" quadratic equations over $\mathbb{F}_q$, i.e., equations of the form $\sum_j h_j x_j^2 = 0$. When there are m equations, this can be regarded as the CIS problem with allowed set $A = \{a^2 : a \in \mathbb{F}_q\}$ the quadratic residues. They gave an efficient algorithm whenever $m \geq (n+1)(n+2)/2$. As noted in Section 1.1, when $q = 3$, this problem is equivalent to $\mathbb{F}_3^n$-Subset-Sum.

The followup work of Ivanyos and Santha [IS17] generalized the problem and algorithm to solving systems of diagonal dth-power equations; this is equivalent to CIS with $A = \{a^d : a \in \mathbb{F}_q\}$. They gave a $\mathrm{poly}(m, \log q)$-time classical algorithm whenever

$$m \geq d^{d(d-1)\lceil \log_2(d+1)\rceil/2}(n+1)^{d\lceil \log_2(d+1)\rceil} \approx d^{d^2 \log d} n^{d \log d}.$$

They also observed that taking $d = q - 1$ gives an $\mathbb{F}_q^n$-Subset-Sum algorithm.

Finally, Imran and Ivanyos [II24] also investigated $\mathbb{F}_q^n$-Subset-Sum and slightly sharpened the result of [IS17], finding solutions in $\mathrm{poly}(m)$ time whenever

$$m \geq q^{Cq \log^2 q} \cdot n^{\lfloor q/2 \rfloor \cdot \lceil \log_2 q \rceil} \approx q^{q \log^2 q} \cdot n^{q \log q}.$$

A central ingredient in their method is what we call the "halving trick". The base version of this trick lets them solve SIS$^\infty$ with $s = \lfloor q/4 \rfloor$ whenever $m \geq Cqn^2$, and they also iterate it to achieve smaller s. In Section 3, we give an improvement on the halving trick that eliminates the q-factor in the m-dependence: it achieves $s = \lfloor q/4 \rfloor$ whenever $m \geq Cn^2$. This means the halving trick can be employed effectively even when q is exponentially large in n. One can get a lot of mileage out of the halving trick, but our strongest results (Section 1.4) are ultimately obtained by abandoning it in favor of a more sophisticated generalization in Section 4.

5.3 Quantum algorithms from Regev's reduction

Aside from the work of Chen, Liu, and Zhandry [CLZ22], at least two other candidate exponential quantum speedups have been derived via Regev's framework of converting an A-CIS-type problem (usually in its equivalent dual version) to a quantum decoding [CT24] problem. These are: the Yamakawa–Zhandry paper [YZ24], which gives a provable exponential quantum speedup in the random oracle query model; and, the DQI paper of Jordan et al. [JSW+24], which gives a seeming exponential quantum speedup for the "OPI" problem of fitting a low-degree univariate polynomial over $\mathbb{F}_q$ to a range of points (see also the followup [CT24]).

It is interesting to understand what aspects of these problems cause them to resist our dequantization efforts. The fact that both problems are "worst-case/structured" — meaning that the input codes are not assumed to be chosen randomly — does not necessarily pose a problem for dequantization, as many of our algorithms work in the worst case. The Yamakawa–Zhandry problem has q exponentially large as a function of n, but this too does not seem to be an inherent problem for classical algorithms, as our algorithms mostly have $\log q$ dependence. The fact that the set A in Yamakawa–Zhandry has no structure — indeed, is random — certainly makes it harder to dequantize. That said, we do not know how to dequantize even the OPI problem when the set A is assumed to be an interval of width $q/2$, the most structured possible A. It seems that the biggest difficulty in dequantizing both Yamakawa–Zhandry and the DQI algorithm for OPI is the fact that $m = O(n)$ for these problems; whereas, our efficient classical algorithms do not seem to be able to get off the ground unless $m \geq \Omega(n^2)$.

5.4 SIS$^\infty$-related problems

Here we mention some problems that are either equivalent to SIS$^\infty$, or very nearly equivalent.

Versus traditional Subset-Sum. $\mathbb{F}_q^n$-Subset-Sum is a variant of the traditional Subset-Sum problem (with target 0), one of the most canonical NP-complete problems. In fact, the textbook reduction from (say) 1-in-3-Sat to Subset-Sum works out much more cleanly for $\mathbb{F}_q^n$-Subset-Sum, for any $q > 2$, since no tricks are needed to prevent carries.[9] We also remark that over integers, the $\{\pm 1\}$-CIS problem is known as Partitioning (one of Garey and Johnson's 6 canonical NP-complete problems) or Pigeonhole-Equal-Sums. It is a "total" problem by the Pigeonhole

[9] The reduction is as follows. Given an m-clause n-variate 1-in-3-Sat instance, produce $2n+1$ vectors in dimension $n+m$. For each literal ℓ_i, make a vector with 1 in the ith coordinate and in the $(i+j)$th coordinate whenever ℓ_i satisfies the jth clause. Finally, include the vector $-(1, \ldots, 1)$. Correctness uses $1 \notin \{0, 2, 3\} \bmod q$.

Principle whenever $2^m > q^n$. This implies that $m > (\log_2 q)n$, and hence the problem is in the TFNP class PPP. We also mention that a variant of SIS$^\infty$ is proven to be PPP-complete [SZZ18].

The equivalent "dual" problem: LWE. We recall the *equivalent*, linear algebraic dual problem to SIS$^\infty$. Given input $H \in \mathbb{F}_q^{n\times m}$, the SIS$^\infty$ problem may be described as seeking a (nonzero) *codeword* x, satisfying $\|x\|_\infty \le s$, in the $\mathbb{F}_q$-linear code whose parity-check matrix is H. The problem is easily interreducible to the version in which the code is presented by a *generator matrix* $G \in \mathbb{F}_q^{k\times m}$, $k := m - n$. Now the task may be described as seeking an unconstrained $z \in \mathbb{F}_q^k$ such that zG is nonzero but "short", $\|zG\|_\infty \le s$.

This problem looks even more familiar if one starts from the more general targeted version of SIS$^\infty$. Then, given H and target t, one is seeking a short codeword x in the *affine* code defined by $Hx = t$. In the generator matrix formulation, this is seeking an unconstrained $z \in \mathbb{F}_q^k$ such that $\|zG - b\|_\infty \le s$ for an arbitrary b satisfying $Hb = t$. Equivalently, this is seeking a mod-q solution $z \in \mathbb{F}_q^k$ to the m "noisy equations"

$$g_i \cdot z \approx b_i, \tag{18}$$

where $\approx$ denotes the ℓ_∞ norm of their difference is at most s. In this (equivalent) formulation, the problem strongly resembles the Learning With Errors (LWE) problem. The main differences are:

- LWE usually assumes one is in the *planted random* case, where a "secret" solution z^* is chosen at random, then the vectors g_i are chosen uniformly at random, and finally each b_i is taken to be the planted value $g_i \cdot z^*$ with random noise of "width" s. The noise is typically discrete-Gaussian of standard deviation s, but uniform on $-s, \dots, s$ is also reasonable.
- In LWE, the parameters are usually chosen so that $m = k^C$ for some (possibly large) constant C. In the (equivalent) targeted SIS$^\infty$ problem, this corresponds to "m" (the number of input vectors) being only slightly larger than "n" (the dimension): $m = n + n^{1/C}$.

Recall that when the input is uniformly random (not planted), there is unlikely to be a solution unless $m \ge (\log_2 q)n$. Thus in the "LWE regime" of $m = n + n^{1/C}$, it only makes sense to study the refutation problem or the planted problem. In the present paper, our SIS$^\infty$ results are only for m at least $\Omega(n^2)$, so we have nothing to say about LWE.

LIN-SAT. The equivalent dual version of CIS is a common viewpoint for the recent quantum algorithms using Regev's reduction. In this viewpoint, the input is an $\mathbb{F}_q$-linear code $\mathcal{C} \subseteq \mathbb{F}_q^m$ of dimension $k = m - n$ together with a subset $A \subseteq \mathbb{F}_q$. The task is to find a codeword $y \in \mathcal{C}$ such that $y_i \in A$ for all $i \in [m]$. In the targeted and even more general version, this would mean $y_i \in b_i + A_i$ for all i, where b is also part of the input.

This form of the problem was called LIN-SAT in [JSW+24]. This viewpoint is adopted in the Yamakawa–Zhandry problem [YZ24], where q is exponentially large in m, $\mathcal{C}$ is an efficiently decodeable, list-recoverable code (specifically, a folded Reed–Solomon code), and $A \subseteq \mathbb{F}_q$ is a *random* set of cardinality roughly $q/2$. It is also the viewpoint in the Optimal Polynomial Interpolation (OPI) problem from [JSW+24], where we work in the targeted version, $q \approx m$, $k \approx m/10$, and $\mathcal{C}$ is the Reed–Solomon code.

The $q = 3$ case. We remark that the $q = 3$ case of $\{\pm 1\}$-CIS is particularly attractive. In its original, primal version, it is asking to partition a sequence of m vectors from $\mathbb{F}_q^n$ into two subsequences of equal sum. In its dual version, it is asking to find a codeword in a given $\mathbb{F}_3$-linear code with all symbols in $\{\pm 1\}$, i.e., at maximal Hamming distance m from the all-zero codeword.

This "ternary syndrome decoding with maximal weight" problem was carefully studied in the context of the Wave cryptosystem [DST19]. To write the problem "LWE"-style, it is equivalent to solving a system of *disequations* over $\mathbb{F}_3$,

$$g_i \cdot s \neq 0. \tag{19}$$

This problem of *learning from disequations*, which also makes sense over fields larger than $\mathbb{F}_q$, has been studied in several prior works, dating back to Friedl et al. [FIM+02], and developed by Ivanyos [Iva07], Arora–Ge [AG11], and Ivanyos–Prakash–Santha [IPS18].

5.5 Zero-sum theory

The following arguably surprising fact dates to the late 1960s.

Fact 5.2 ([Ols69, vEB69])**.** Let q be a prime. Given sequence $h_1, \ldots, h_m \in \mathbb{F}_q^n$, there is always a nonempty subsequence summing to 0 provided $m > (q-1)n$ (and this bound is tight).

In other words, for $m > (q-1)n$ the $\mathbb{F}_q^n$-Subset-Sum problem (and hence SIS^∞ for any s) is "total": there is always a solution. Unlike other similar constraint satisfaction problems, this puts $\mathbb{F}_q^n$-Subset-Sum into the "TFNP" framework; in particular, for $m > (q-1)n$ the task is PPA_q [GKSZ20, SZZ18].

Fact (5.2) holds more generally for p-groups, and is one of the seminal results in the field of *zero-sum theory*. For a survey of this research area, which investigates Subset-Sum and related problems like SIS^∞ with $s = 1$ over abelian groups, see e.g., [GG06].

The original proof of Fact (5.2) was via an exponential-time algorithm. These days it is common to derive it from the Chevalley–Warning Theorem, or more generally Combinatorial Nullstellensatz. For interested readers, we record here a streamlined proof.

Proof of Fact (5.2). Recall that $x \mapsto x^{q-1}$ maps zero/nonzero to 0/1 in $\mathbb{F}_q$. Thus we need to find nonzero $x \in \mathbb{F}_q^m$ such that $\mathrm{EQ}_i(x) := \sum_{j=1}^m h_j[i] x_j^{q-1} = 0$ for $i = 1 \ldots n$, where $h_j[i]$ is the ith coordinate of h_j. We encode the conditions with polynomials:

$$\mathrm{SAT}(x) := (1 - \mathrm{EQ}_1(x)^{q-1})(1 - \mathrm{EQ}_2(x)^{q-1}) \cdots (1 - \mathrm{EQ}_n(x)^{q-1}) \in \{0, 1\} \tag{20}$$

signifies whether x gives a zero-sum, and

$$Z(x) := (1 - x_1^{q-1})(1 - x_2^{q-1}) \cdots (1 - x_m^{q-1}) \in \{0, 1\} \tag{21}$$

detects whether $x = 0$. Thus we seek any $x \in \mathbb{F}_q^m$ such that $\mathrm{OK}(x) := \mathrm{SAT}(x) - Z(x) \neq 0$. Recall that every function $f : \mathbb{F}_q^m \to \mathbb{F}_q$ has a unique representation as a polynomial with individual degree at most $q-1$, obtained by taking any interpolating polynomial and reducing via $x_i^q \mapsto x_i$. But even after reduction, $\mathrm{OK}(x)$ will have a term $\pm x_1^{q-1} \cdots x_m^{q-1}$ of degree $(q-1)m$ coming from $Z(x)$. This cannot be canceled out from $\mathrm{SAT}(x)$ because SAT has degree at most $(q-1)^2 n$, which is strictly less than $(q-1)m$ by assumption. Thus $\mathrm{OK} : \mathbb{F}_q^m \to \mathbb{F}_q$ has unique polynomial representation which is nonzero, and is thus a nonzero function, i.e., there is at least one $x \in \mathbb{F}_q^m$ with $\mathrm{OK}(x) \neq 0$. □

5.6 Post-quantum cryptosystem

We discuss two recent proposals for NIST post-quantum cryptosystems whose security relies on the hardness of variants of SIS^∞. We stress that both cryptosystems have $m \approx 2n$, whereas our algorithms require $m \geq \Omega(n^2)$.

The *Wave* signature scheme [DST19, BCC+22] is essentially based on the assumed hardness of average-case $\{\pm 1\}$-CIS over $\mathbb{F}_3^n$, which is equivalent in hardness to average-case $\mathbb{F}_3^n$-Subset-Sum. More precisely, it is based on a version with: (i) a planted trapdoor; (ii) a relaxation that only "most" coordinates of the solution need be in $\{\pm 1\}$. Nevertheless, the most stringent cryptanalysis of the system [BCDL19] focused on the basic average-case CIS problem. The recommended parameter setting for optimal security was roughly $m \approx 2n$, and [BCDL19] presented convincing evidence that a break required $2^{\Omega(n)}$ time. For $m \gg n$, a Blum–Kalai–Wasserman/Wagner-style algorithm [BKW03, Wag02] will solve the problem in time $\exp(O(n/\log(m/n)))$. But it is interesting to speculate if this can be improved for $m = n^c$ where $1 < c < 2$. We know from our Theorem 1.2 that the task becomes easy once $m \geq n^2/3$.

The [DKL+18, LDK+] CRYSTALS-Dilithium signature scheme was selected for standardization by NIST's post-quantum cryptography initiative. One of the key hardness assumptions underlying its security is "Module-SIS$^\infty$", which is a variant of SIS$^\infty$. A concrete, comparable version of SIS$^\infty$ that is presumed intractable has $q \approx 2^{23}$, $s \sim q/8$, and $m \sim 1.9n$, with $n = 1280$. Our SIS$^\infty$ algorithms are successful at handling $q \gg n$, but require $m \geq Cn^4$ in order to achieve $s \sim q/8$. Again, it would be interesting to investigate subexponential-time tradeoffs that interpolate between the polynomial-time-solvable case of $m \approx n^4$ and the presumed exponential-time case of $m \approx n$.

Acknowledgments

We are very grateful to David Gosset for his contributions to the early stages of this work. RO thanks Siddhartha Jain for helpful discussions. KW thanks Ce Jin, Qipeng Liu, and Hongxun Wu for relevant references. We acknowledge the use of Gemini and ChatGPT to search the literature and suggest proof strategies.

KW is supported by the National Science Foundation under Grant No. DMS-2424441, and by the IAS School of Mathematics.

References

[ADW+25] Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2005–2039. SIAM, 2025. [p. 30]

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In *International Colloquium on Automata, Languages, and Programming*, pages 403–415. Springer, 2011. [pp. 5, 34]

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems. In *Symposium on the Theory of Computing*, pages 99–108. ACM, New York, 1996. [p. 5]

[BCC+22] Gustavo Benegas, Kévin Carrier, André Chailloux, Alain Couvreur, Thomas Debris-Alazard, Phillippe Gaborit, Pierre Karpman, Johanna Loyer, Ruben Niederhagen, Nicolas Sendrier, Benjamin Smith, and Jean-Pierre Tillich. Wave round one submission. Technical report, NIST, 2022. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/wave-spec-web.pdf. [p. 35]

[BCDL19] Rémi Bricout, André Chailloux, Thomas Debris-Alazard, and Matthieu Lequesne. Ternary syndrome decoding with large weight. Cryptology ePrint Archive, Paper 2019/304, 2019. URL: https://eprint.iacr.org/2019/304. [p. 35]

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, 50(4):506–519, 2003. [p. 35]

[CLZ22] Yilei Chen, Qipeng Liu, and Mark Zhandry. Quantum algorithms for variants of average-case lattice problems via filtering. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 372–401. Springer, 2022. [pp. 3, 4, 5, 6, 7, 8, 27, 31, 32]

[CT23] André Chailloux and Jean-Pierre Tillich. The quantum decoding problem. *arXiv preprint arXiv:2310.20651*, 2023. [p. 3]

[CT24] André Chailloux and Jean-Pierre Tillich. Quantum advantage from soft decoders. *arXiv preprint arXiv:2411.12553*, 2024. [pp. 3, 32]

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 238–268, 2018. [pp. 3, 35]

[DST19] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 21–51. Springer, 2019. [pp. 3, 34, 35]

[ET36] Paul Erdős and Paul Turán. On some sequences of integers. *Journal of the London Mathematical Society*, 1(4):261–264, 1936. [p. 27]

[FIM+02] Katalin Friedl, Gábor Ivanyos, Frédéric Magniez, Miklos Santha, and Pranab Sen. Hidden translation and translating coset in quantum computing. Technical report, arXiv quant-ph/0211091, 2002. [p. 34]

[GG06] Weidong Gao and Alfred Geroldinger. Zero-sum problems in finite abelian groups: a survey. *Expositiones Mathematicae*, 24(4):337–369, 2006. doi:10.1016/j.exmath.2006.07.002. [p. 34]

[GKSZ20] Mika Göös, Pritish Kamath, Katerina Sotiraki, and Manolis Zampetakis. On the complexity of modulo-q arguments and the Chevalley–Warning theorem. In *35th Computational Complexity Conference*, volume 169 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 19, 42. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2020. [p. 34]

[II24] Muhammad Imran and Gábor Ivanyos. Zero sum subsequences and hidden subgroups. *Quantum Information Processing*, 23(1):14, 2024. [pp. 4, 6, 8, 10, 14, 16, 31, 32]

[IPS18] Gábor Ivanyos, Anupam Prakash, and Miklos Santha. On learning linear functions from subset and its applications in quantum computing. Technical Report 1806.09660, 2018. [p. 34]

[IS17] Gábor Ivanyos and Miklos Santha. Solving systems of diagonal polynomial equations over finite fields. *Theoretical Computer Science*, 657(Part A):73–85, 2017. doi:10.1016/j.tcs.2016.04.045. [pp. 31, 32]

[ISS07] Gábor Ivanyos, Luc Sanselme, and Miklos Santha. An efficient quantum algorithm for the hidden subgroup problem in nil-2 groups. *Algorithmica*, 62(1):480–498, 2012 (preprint on ArXiv 2007). [pp. 5, 8, 9, 31]

[Iva07] Gábor Ivanyos. On solving systems of random linear disequations. Technical report, arXiv:0704.2988, 2007. [p. 34]

[JSW+24] Stephen P Jordan, Noah Shutty, Mary Wootters, Adam Zalcman, Alexander Schmidhuber, Robbie King, Sergei V Isakov, and Ryan Babbush. Optimization by decoded quantum interferometry. *arXiv preprint arXiv:2408.08292*, 2024. [pp. 3, 7, 32, 33]

[KM23] Zander Kelley and Raghu Meka. Strong bounds for 3-progressions. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 933–973. IEEE, 2023. [pp. 27, 38]

[LDK+] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-DILITHIUM. https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. [p. 35]

[Lev00] Vsevolod Lev. Simultaneous approximations and covering by arithmetic progressions in $\mathbb{F}_p$. *Journal of Combinatorial Theory. Series A*, 92(2):103–118, 2000. doi:10.1006/jcta.1999.3034. [pp. 17, 38]

[LSS24] James Leng, Ashwin Sah, and Mehtaab Sawhney. Improved bounds for szemerédi's theorem. *arXiv preprint arXiv:2402.17995*, 2024. [pp. 27, 38]

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM journal on computing*, 37(1):267–302, 2007. [p. 5]

[Ols69] John Olson. A combinatorial problem on finite Abelian groups. I. *Journal of Number Theory*, 1:8–10, 1969. doi:10.1016/0022-314X(69)90021-3. [p. 34]

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009. [p. 3]

[Ste24] Matthias Steiner. The complexity of algebraic algorithms for LWE. In *EUROCRYPT 2024*, volume 14653 of *Lecture Notes in Computer Science*, pages 375–403. Springer, 2024. [p. 5]

[SZZ18] Katerina Sotiraki, Manolis Zampetakis, and Giorgos Zirdelis. PPP-completeness with connections to cryptography. In *59th Annual IEEE Symposium on Foundations of Computer Science*, pages 148–158. IEEE, 2018. [pp. 33, 34]

[vEB69] Peter van Emde Boas. A combinatorial problem on finite abelian groups. II. *Mathematisch Centrum Amsterdam. Afdeling Zuivere Wiskunde*, 1969(ZW-007):60, 1969. [p. 34]

[Wag02] David Wagner. A generalized birthday problem. In *Annual International Cryptology Conference*, pages 288–304. Springer, 2002. [p. 35]

[YZ24] Takashi Yamakawa and Mark Zhandry. Verifiable quantum advantage without structure. *Journal of the ACM*, 71(3):1–50, 2024. [pp. 3, 7, 32, 33]

A *Ad hoc* arithmetic combinatorics

We prove the arithmetic combinatorial results here. While significantly better bounds are known [KM23, LSS24] using more sophisticated techniques, we choose to provide what we need with minimal efforts as this is not the focus of our work.

We first prove Fact (3.16).

Fact 3.16 ([Lev00])**.** Let $q \geq 5$ be a prime. Let $A \subseteq \mathbb{F}_q$ be an arbitrary set of size $|A| \geq q - \log_4(q+2)$. Then A contains an AP of length $(q+1)/2$.

Proof. Let $\overline{A} = \mathbb{F}_q \setminus A = \{a_1, \ldots, a_c\}$ where $c = |\overline{A}| \leq \log_4(q+2)$. By translation, we assume $0 \notin \overline{A}$. Partition $\mathbb{F}_q \setminus \{0\}$ into 4 contiguous intervals I_1, I_2, I_3, I_4, each of length at most $\lceil (q-1)/4 \rceil$.

For each $s \in \mathbb{F}_q \setminus \{0\}$, define $v^{(s)} \in \{0,1,2,3\}^c$ where the ith entry of $v^{(s)}$ indicates the interval that $s \cdot a_i$ lies in, i.e., $v^{(s)}[i] = b \in \{0,1,2,3\}$ if $s \cdot a_i \in I_b$. Now we divide into the following cases.

- If for some s and $b \in \{0,1,2\}$ we have $v^{(s)}[i] = b$ for all $i \in [c]$, then $s\overline{A}$ is contained in I_b, which is an interval of length $|I_b| \leq \lceil (q-1)/4 \rceil \leq (q-1)/2$. This means sA contains an interval of length at least $(q+1)/2$, which is an AP of the same length in A after dilation.
- The above cases correspond to 4 patterns in $\{0,1,2,3\}^c$. If none of them appear, it remains $4^c - 4$ possible patterns. On the other hand, we have $q-1$ choices of s. Since $q - 1 \geq 4^c - 3$, there exist distinct $s, s' \in \mathbb{F}_q \setminus \{0\}$ such that $v^{(s)} = v^{(s')}$. Then for $\overline{s} = s - s' \neq 0$, each $\overline{s} \cdot a_i = s \cdot a_i - s' \cdot a_i \in (I_b - I_b) \setminus \{0\}$ for some $b \in \{1,2,3,4\}$. Since each I_b is an interval of length at most $\lceil (q-1)/4 \rceil$, every element in $I_b - I_b$ has absolute value at most $\lceil (q-1)/4 \rceil - 1 \leq (q-1)/4 - 1/2$ since $q - 1 \pmod 4 \in \{0,2\}$. This means $\overline{s}A$ contains the interval of length $q - 2\left((q-1)/4 - 1/2\right) - 1 = (q+1)/2$, which is an AP of the same length in A after dilation. □

Now we turn to Fact (4.16) and Fact (4.17).

Fact 4.16. Let $q \geq 3$ be a prime. Let $A \subseteq \mathbb{F}_q$ and define $c = |\mathbb{F}_q \setminus A|$. There exist $x, y, z \in \mathbb{F}_q$ such that the following holds.

1. If $2 \leq c < q$, then $x \neq 0$, $z \in A$, and $z - x, z + x \notin A$.
2. In addition to the conditions in Item 1, if $c < (q+1)/2$, then $y \neq 0$ and $z - y, z + y \in A$.

Proof. Define $\overline{A} = \mathbb{F}_q \setminus A$. For distinct $u, v \in \overline{A}$, let $z(u,v) = \frac{u+v}{2}$, which is well-defined as $q \geq 3$. Consider $Z = \{z(u,v) \colon u, v \in \overline{A} \text{ and } u \neq v\}$. We will show that $Z \cap A \neq \emptyset$.

For each $z \in \overline{A}$, we have

$$|\{(u,v) \colon u, v \in \overline{A} \text{ and } u \neq v \text{ and } z(u,v) = z\}| \leq 2\lfloor (c-1)/2 \rfloor, \tag{22}$$

since $u, z(u,v), v$ is a nontrivial 3-AP in $\overline{A}$. Hence

$$P := \sum_{z \in \overline{A}} |\{(u,v) \colon u, v \in \overline{A} \text{ and } u \neq v \text{ and } z(u,v) = z\}| \leq 2c\lfloor (c-1)/2 \rfloor.$$

On the other hand, we have

$$Q := \sum_{z \in Z} |\{(u,v) \colon u,v \in \overline{A} \text{ and } u \neq v \text{ and } z(u,v) = z\}| = |\{(u,v) \colon u,v \in \overline{A} \text{ and } u \neq v\}| = c(c-1).$$

Assume towards the contradiction that $Z \setminus \overline{A} = Z \cap A = \emptyset$. Then $Z \subseteq \overline{A}$ and hence $P \geq Q$.

- If c is even, then $Q = c(c-1) > 2c\lfloor (c-1)/2 \rfloor = P$. A contradiction.
- If c is odd, then $Q = c(c-1) = 2c\lfloor (c-1)/2 \rfloor = P$. This means every inequality in Equation (22) is an equality. That is, for every $z \in \overline{A}$, the set $\overline{A} \setminus \{z\}$ is formed by $(c-1)/2$ nontrivial antipodal pairs $\{u,v\}$ with the common center z. Note that each such a pair $\{u,v\}$ satisfies $u + v = 2z$. Hence $\sum_{u \in \overline{A}} u = (1 + 2 \cdot (c-1)/2) \cdot z = c \cdot z$ holds for any $z \in \overline{A}$, which means $c \cdot z = c \cdot z'$ for any $z, z' \in \overline{A}$. Since $2 \leq |\overline{A}| = c < q$, this is impossible.

Now we fix an arbitrary $z \in Z \cap A$ and arbitrary $u \neq v \in \overline{A}$ satisfying $z(u,v) = z$. Define $x = z - u$. Then Item 1 immediately holds.

To find y satisfying Item 2, we observe that $\{z-x, z+x\} \subseteq \overline{A} \subseteq \mathbb{F}_q \setminus \{z\}$ by Equation (22). Since $q \geq 3$, we know that $\mathbb{F}_q \setminus \{z-x, z, z+x\}$ is a disjoint partition of $(q-3)/2$ antipodal pairs $\{z-y, z+y\}$ for $y \notin \{0, \pm x\}$. As $\left|\overline{A} \setminus \{z-x, z+x\}\right| = c - 2 < (q-3)/2$, there exists a pair $\{z-y, z+y\}$ for some $y \notin \{0, \pm x\}$ that is not contained in $\overline{A}$, i.e., $z - y, z + y \in A$ as desired. □

To handle the edge case that $c = (q+1)/2$ and prove Fact (4.17), we need the following simple fact.

Fact A.1. Let $c_{-4}, c_{-3}, \ldots, c_4 \in \{0,1\}$ be arbitrary satisfying $c_0 = 0$ and $c_i + c_{-i} = 1$ for $i = 1, 2, 3, 4$. Then there always exist $-4 \leq j < k < \ell \leq 4$ such that $j + \ell = 2k$ and $c_j = c_k = c_\ell = 0$.

Proof. Assume towards the contradiction that such j, k, ℓ do not exist.

By symmetry, we fix $c_1 = 0, c_{-1} = 1$. To avoid $c_0 = c_1 = c_2 = 0$, we must have $c_{-2} = 0, c_2 = 1$. To avoid $c_{-2} = c_1 = c_4 = 0$, we now must have $c_{-4} = 0, c_4 = 1$. However at this point we cannot avoid $c_{-4} = c_{-2} = c_0 = 0$, which is a contradiction. □

Now we prove Fact (4.17).

Fact 4.17. Let $q \geq 11$ be a prime. Let $A \subseteq \mathbb{F}_q$ and define $c = |\mathbb{F}_q \setminus A|$. Assume $c = (q+1)/2$. Then A contains a 3-AP.

Proof. Since $c = (q+1)/2$ and $q \geq 11$, we know $2 \leq c < q$. By Item 1 of Fact (4.16), we can find $x \neq 0$ and $z \in A$ such that $z - x, z + x \notin A$.

Define $m = \lfloor q/2 \rfloor$ and $t = m/x \in \mathbb{F}_q$. Consider $A' = t(A - z)$. Then $0 \in A'$ and $\pm m \notin A'$. In addition, any 3-AP in A' corresponds to a 3-AP in A by translation and dilation. Hence it suffices to find a 3-AP in A'.

Observe that $0 \in A' \subseteq \mathbb{F}_q \setminus \{\pm m\}$. Since $q > 2$, we know that $\mathbb{F}_q \setminus \{\pm m\}$ is a disjoint partition of $(q-3)/2$ antipodal pairs $\{\pm y\}$ for $y \notin \{0, \pm m\}$. Define $\overline{A'} = \mathbb{F}_q \setminus A'$. As $\left|\overline{A'} \setminus \{\pm m\}\right| = c - 2 = (q-3)/2$, we run into one of the following two cases.

- There exists a pair $\{\pm y\}$ for some $y \notin \{0, \pm m\}$ that is not contained in $\overline{A'}$.

  Then we observe that $-y, 0, y \in A'$ is a 3-AP.

- For every $y \notin \{0, \pm m\}$, exactly one element of the pair $\{\pm y\}$ is contained in $\overline{A'}$.

  Since $q \geq 11$ and $m = \lfloor q/2 \rfloor \geq 5$, we now focus on the range $i = -4, -3, \ldots, 4 \in \mathbb{F}_q$ of 9 elements. For each i in the range, define $c_i = 0$ if $i \in A'$; and $c_i = 1$ if $i \in \overline{A'}$. Then we have $c_0 = 0$ and $c_i + c_{-i} = 1$. By Fact (A.1), there exist $-4 \leq j < k < \ell \leq 4$ such that $j + \ell = 2k$ and $c_j = c_k = c_\ell = 0$, which means $j, k, \ell \in A'$. Hence j, k, ℓ is a 3-AP in A'.

For both cases, we find a 3-AP in A' (and thus A), which completes the proof of Fact (4.17). □

B Fast basis search

We use fast matrix multiplication to obtain fast algorithms to find basis vectors and prove Fact (5.1).

We will need the following subroutine.

Fact B.1. There is a deterministic algorithm, given any $1 \leq \ell \leq 2n$ vectors in $\mathbb{F}_q^n$ and running in time $\ell^{\omega-1} n \cdot \mathrm{polylog}(n, q)$, that outputs a maximal linearly independent set among the vectors.

Proof. Let $T(\ell)$ be the runtime upper bound and apparently $T(1) = n \cdot \mathrm{polylog}(n, q)$ as claimed.

The algorithm proceeds in a recursive fashion for $\ell \geq 2$: we partition the ℓ vectors into two parts with $\ell_1 = \lfloor \ell/2 \rfloor$ and $\ell_2 = \lceil \ell/2 \rceil$ vectors each, denoted $v_1, \ldots, v_{\ell_1}$ and $u_1, \ldots, u_{\ell_2}$ respectively. We recursively find a maximal linearly independent set of size $t \leq \ell_1$, stacked as $V \in \mathbb{F}^{n \times t}$, from the first part. Then we project $u_1, \ldots, u_{\ell_2}$ onto the orthogonal space spanned by columns of V; and find a maximal linearly independent set there. The final output is by merging all the vectors found above.

To analyze the runtime, recall that the projector onto the column space of V is defined as $P = V(V^\top V)^{-1} V^\top$. Define $U = \begin{bmatrix} u_1 & \cdots & u_{\ell_2} \end{bmatrix}$. Then the projection of $u_1, \ldots, u_{\ell_2}$ is equivalent to computing $U - PU = U - V(V^\top V)^{-1} V^\top U$. Since $V \in \mathbb{F}^{n \times t}, t \leq \ell_1 = \lfloor \ell/2 \rfloor \leq n$ and $U \in \mathbb{F}^{n \times \ell_2}, \ell_2 \leq \lceil \ell/2 \rceil \leq n$, the computation of $A = V^\top V$ and $B = V^\top U$ takes time $\ell^{\omega-1} n \cdot \mathrm{polylog}(n, q)$ by partitioning U, V into square blocks and combining $O(n/\ell)$ multiplications of $O(\ell)$ by $O(\ell)$ matrices. Then A^{-1} takes time $\ell^\omega \cdot \mathrm{polylog}(n, q) \leq \ell^{\omega-1} n \cdot \mathrm{polylog}(n, q)$. Finally $VA^{-1}B$ takes time $\ell^{\omega-1} n \cdot \mathrm{polylog}(n, q)$ by a similar block matrix multiplication fashion. Hence the runtime $T(\ell)$ satisfies the following recursion

$$\begin{aligned} T(\ell) = & \underbrace{T(\ell_1)}_{\text{first recursion}} + \underbrace{\ell^{\omega-1} n \cdot \mathrm{polylog}(n, q)}_{\text{projection}} + \underbrace{T(\ell_2)}_{\text{second recursion}} + \underbrace{\ell n \cdot \mathrm{polylog}(n, q)}_{\text{merge}} \\ & = \mathrm{polylog}(n, q) \cdot \left(\sum_{0 \leq i \leq \log(\ell)} 2^i \cdot (\ell/2^i)^{\omega-1} \cdot n \right) \\ & = \ell^{\omega-1} n \log(n) \cdot \mathrm{polylog}(n, q) = \ell^{\omega-1} n \cdot \mathrm{polylog}(n, q) \qquad (\text{since } \omega \geq 2 \text{ and } \ell \leq 2n) \end{aligned}$$

as desired. □

Now we are ready to prove Fact (5.1).

Fact 5.1. There is a deterministic algorithm, given any $m \geq n$ vectors in $\mathbb{F}_q^n$ and running in time $n^{\omega-1} m \cdot \mathrm{polylog}(n, q)$, that outputs a maximal linearly independent set among the vectors.

Proof. We divide the m input vectors into $k = \lceil m/n \rceil$ batches, where each batch contains at most n vectors. We maintain a set S of linearly independent vectors and go over each batch to update S. Since each batch has at most n vectors and $|S| \leq n$, each update on S is equivalent to finding a basis for at most $2n$ vectors in $\mathbb{F}^n$, which has runtime $n^\omega \cdot \mathrm{polylog}(n, q)$ by Fact (B.1). Hence the total runtime is $n^\omega k \cdot \mathrm{polylog}(n, q) = n^{\omega-1} m \cdot \mathrm{polylog}(n, q)$. □