

Two-Bit Lifting for Ternary SIS: Polynomial-Time Collision Attacks on LtHash

Rui Ding^{1,2}, Xiaorui Gong^{1,2}, Hao Jiang^{1,2}, and Lili Tang^{1,2}

¹ Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

² School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{dingrui2024, gongxiaorui, haojiang, tanglili}@iie.ac.cn

Abstract. LtHash is a lattice-based incremental hash function introduced by Bellare and Micciancio at EUROCRYPT 1997. It can be used in distributed systems to efficiently maintain digests of changing data. Its set-collision resistance rests on the average-case hardness of the ternary short integer solution (SIS) problem in the random oracle model. LtHash implementations typically use power-of-two groups $(\mathbb{Z}/2^d\mathbb{Z})^N$ for efficient machine-word arithmetic and SIMD parallelism. Previous security estimates were based on Wagner-type generalized birthday attacks with sub-exponential running time.

Chen, Liu, and Zhandry (CLZ, EUROCRYPT’22) describe a folklore algorithm attributed to Regev that solves ternary SIS modulo 2^d . Applying this algorithm to LtHash yields polynomial-time collision attacks requiring $O(N^d)$ hash queries, a result that has not been previously documented. We improve the Regev–CLZ algorithm by canceling two modulus bits per layer instead of one, while preserving ternary coefficients. This reduces the number of input columns — and hence the number of LtHash queries — from $O(N^d)$ to $O(N^{\lceil d/2 \rceil})$. This refutes the uniform average-case formulation of Liyan Chen et al.’s conjecture (ITC’26) that the Regev–CLZ bound is optimal. Our technique also extends to inhomogeneous SIS with the same bound.

For LtHash16 with $(N, d) = (1024, 16)$, used at Facebook and in Solana, our attack requires approximately 2^{81} hash queries. Its expected computational cost is at most 2^{101} scalar operations over $\mathbb{F}_2$, compared with our estimates of approximately 2^{180} operations for the Regev–CLZ algorithm and 2^{188} for the state-of-the-art generalized birthday attack of Tang et al. (CRYPTO’26), under their respective operation models. Our query and scalar-operation costs are both far below the level suggested by the original claim of at least 200-bit security.

Keywords: Ternary SIS · Homomorphic hashing · Incremental hashing · LtHash · Collision attacks

1 Introduction

Bellare and Micciancio proposed *LtHASH* (short for lattice hash) at EUROCRYPT 1997 as an incremental hash function operating on indexed block sequences [2]. Its additive structure allows updates in time proportional to the

number of modified blocks. Unlike discrete-logarithm-based alternatives such as MuHash and ECMH [2,17], LtHASH is based on lattice hardness, making it a natural post-quantum candidate.

In 2019, Lewi et al. at Facebook revisited this design for secure update propagation [16], deploying LtHash and releasing its implementation in Facebook’s Folly library [15]. While LtHash supports arbitrary moduli, their instantiation hashes finite sets of bit strings to vectors modulo a power of two, enabling efficient word-level and SIMD arithmetic [16]:

$$\text{LtHash}_{N,d}(S) = \sum_{x \in S} H(x) \in (\mathbb{Z}/2^d\mathbb{Z})^N,$$

where H is modeled as a random oracle. A set collision consists of two distinct finite sets with the same LtHash value. The attacker chooses both sets, with no prescribed bound on their sizes [16]. The recommended instance in [16], *LtHash16*, sets $(N, d) = (1024, 16)$, with a 2 KiB accumulator suited to 16-bit vector arithmetic. Such instantiations over the power-of-two modulus are also adopted by incremental SHAKE [22], which retains the original indexed-block design.

Widespread Adoption. All deployed and released instantiations in Table 1 use power-of-two moduli. Within this family, LtHash16 has become a de facto standard. Adoption has accelerated since 2025, particularly for blockchain state commitments. Folly and Commonware repeat Lewi et al.’s estimate of at least 200 bits of collision security [16,9,20], and Solana cites the same analysis in support of its 128-bit security target [27]. These deployments thus inherit Lewi et al.’s parameters and security analysis.

Table 1. Deployments and libraries using LtHash.

System	Use	(N, d)	Status	Year
<i>Application data</i>				
Facebook/Folly [15]	Configuration updates	(1024, 16)	Deployed	2019
WhatsApp [33]	MAC-protected app state	(64, 16)	Deployed	2026
atproto [3,10,30]	Permissioned repositories	(1024, 16)	Released	2026
Nym [25]	Node directories	(1024, 16)	Released	2026
<i>Blockchain state</i>				
Solana [27]	Account state	(1024, 16)	Deployed	2025
Sei [28]	Key-value state	(1024, 16)	Released	2026
Vextaris [31]	State roots	(1024, 16)	Testnet	2026
misakas [24]	UTXO commitments	(1024, 32)	Testnet	2026
BNB Chain [4]	Storage for a new L1	Unspecified	Planned	2026
<i>Libraries</i>				
Commonware [9]	Set hashing library	(1024, 16)	Released	2026

Security analyses of power-of-two LtHash have focused on Wagner-type generalized birthday algorithms with sub-exponential running time [16,29]. Lewi et al. evaluated the generalized birthday methods of Camion–Patarin–Wagner (CPW) and Minder–Sinclair (MS) for LtHash16 [16]. Tang, Sun, and Gong (TSG) subsequently applied their algorithms for the Loose Generalized Birthday Problem to indexed-block hashing, giving the state-of-the-art generalized birthday attack [29].

However, while a power-of-two modulus enables efficient arithmetic, it also introduces rich algebraic structure that, as our analysis shows, can be exploited algorithmically. Our new attack based on solving ternary SIS uses about $2^{81.07}$ queries to H and fewer than $2^{100.95}$ scalar operations over $\mathbb{F}_2$ in expectation (Table 2). The LtHash16 primitive does not meet its claimed 200-bit security target, nor even a reduced 128-bit security target.

Set Collision Resistance and Ternary SIS. Recall that messages that are finite sets³ of elements from $\{0, 1\}^*$, are hashed as $\text{LtHash}_{N,d}(M) = \sum_{x \in M} H(x) \pmod{2^d}$, where $H : \{0, 1\}^* \rightarrow (\mathbb{Z}/2^d\mathbb{Z})^N$ is modeled as a random oracle. For distinct queried elements $x_1, \dots, x_m$, let $A = (H(x_1) \mid \dots \mid H(x_m)) \in (\mathbb{Z}/2^d\mathbb{Z})^{N \times m}$. A collision between distinct messages $M_0, M_1 \subseteq \{x_1, \dots, x_m\}$ yields the difference z of their indicator vectors, satisfying

$$z \in \{-1, 0, 1\}^m \setminus \{0\}, \quad Az = 0 \pmod{2^d}.$$

Conversely, any such z gives two distinct colliding messages $M_+ = \{x_i : z_i = 1\}$ and $M_- = \{x_i : z_i = -1\}$. Since the columns of A are independent and uniform, finding collisions between these set-valued messages is equivalent to solving average-case ternary SIS, with one matrix column per distinct oracle query [16].

For prime q , Imran and Ivanyos find ternary relations by repeatedly halving the coefficient bound. Their deterministic algorithm uses $q^{O(q \log q)} N^{O(q)}$ columns and runs in time polynomial in this bound [13]. Kothari, O’Donnell, and Wu reduce the column count for the first halving step from $O(qN^2)$ to $O(N^2)$. They also extend the method to larger reduction factors [14]. These algorithms work for every input matrix. For ternary relations, the degree of their column bounds still grows with the modulus.

For power-of-two moduli, a simple recursive algorithm reduces the modulus from 2^k to 2^{k-1} at each step, thereby eliminating one bit constraint per recursion level. This folklore construction appears in Göös et al. [12] and is attributed to Regev by Chen, Liu, and Zhandry [8, Appendix A]. It finds a nonzero binary relation from any $(N+1)^d$ columns, in polynomial time for fixed d .

We apply the Regev–CLZ recursion to obtain polynomial-time collision attacks, with one hash query per input column. To the best of our knowledge,

³ Unlike conventional hash collisions, where the input is a sequence of message blocks and repeated blocks are meaningful, set collisions operate on sets and therefore ignore multiplicities. Thus, for example, (x, x, y, z) is equivalent to (x, y, z) .

this application has not previously been documented in the hash literature. For LtHash16, our evaluation places this recursion below Lewi et al.’s 200-bit estimate, but still above the 128-bit security target (Table 2).

The $O(N^d)$ column bound is also central to a conjecture of Liyan Chen et al. Their Conjecture B.2 posits polynomial-time hardness for ternary SIS modulo 2^d below $N^{d-\varepsilon}$ columns for fixed $d \geq 2$ and $\varepsilon > 0$ [7, full version, Appendix B]. We examine this conjecture under the uniform average-case formulation (Section 2.3). A natural way to reduce this bound is to eliminate two bits at each recursion step. However, applying this idea naively increases the per-step coefficient growth factor to 2, which compounds exponentially with the recursion depth. The challenge is to keep this factor at one.

Main Results. We improve the Regev–CLZ recursion through two-bit lifting for ternary SIS. The number of input columns drops from $O(N^d)$ to $O(N^{\lceil d/2 \rceil})$, and the resulting LtHash16 collision attack falls below the 128-bit security target. The core idea of our two-bit lifting technique is that changing a sign preserves parity, since $-1 \equiv 1 \pmod{2}$. We first eliminate the first bit and obtain a $\{0, 1\}$ -valued solution vector. We then perform a conditional sign-flipping step on all coordinates with value 1, which eliminates the second bit. In this way, the solution vector still lies in $\{-1, 0, 1\}$.

Table 2. Collision costs for LtHash_{1024,16}.

Method	Queries	Time [‡]	Peak space (bits)
<i>Generalized birthday methods</i>			
CPW [5,32]	2^{167}	2^{263}	$2^{262.44\dagger}$
MS [23]	2^{133}	2^{255}	$2^{255.42\dagger}$
TSG [29] [†]	$2^{181.04}$	$2^{187.54}$	$2^{195.04}$
<i>One-bit recursion</i>			
Regev–CLZ [8, Appendix A] [†]	$2^{160.02}$	$2^{180.03}$	$2^{160.03}$
This work			
Expected	$\approx 2^{81.07}$	$< 2^{100.95}$	$< 2^{88.45}$
Capped run ($p \geq 1/2$)	$< 2^{83.07}$	$< 2^{102.95}$	$< 2^{89.66}$

Costs for prior methods are estimates. Capped bounds hold simultaneously.

[†] Our evaluation. [‡] Method-specific operations (scalar $\mathbb{F}_2$ for both recursions).

Concrete LtHash Bounds. For LtHash16, budgets of $2^{83.07}$ queries and $2^{102.95}$ scalar operations suffice with success probability at least $1/2$, substantially lowering the previous best estimate of approximately 180 bits (Table 2). Our bounds also place *LtHash20* below 128 bits and *LtHash32* below 192 bits (Table 4).

Ternary SIS. We give a classical polynomial-time algorithm for ternary SIS modulo 2^d , using $m = O_d(N^{\lceil d/2 \rceil})$ columns for every fixed $d \geq 2$. For uniform $A \in (\mathbb{Z}/2^d\mathbb{Z})^{N \times m}$, it returns a nonzero $z \in \{-1, 0, 1\}^m$ satisfying $Az = 0$ with probability at least $3/4$ (Theorem 2). This refutes the uniform average-case formulation of Liyan Chen et al.’s Conjecture B.2. For $d = 2$, this conjecture asserts that the lightweight hash $f_A(x) = Ax$ over $\mathbb{Z}/4\mathbb{Z}$, for $x \in \{0, 1\}^m$, is collision resistant when $m \leq N^{2-\varepsilon}$ [7, full version, Appendix B]. Our $O(N)$ -column bound yields a polynomial-time collision attack throughout the super-linear regime $m = \omega(N)$, improving on the $(N+2)^2$ -column bound obtained from Kothari, O’Donnell, and Wu’s ring extension [14, Theorem 4.11 and Section 5.1]. At modulus four, our column bound is optimal up to a constant factor.

Ternary ISIS. We extend our two-bit recursion to ternary inhomogeneous SIS (ISIS). The resulting algorithm solves $Az = b$ for every target b fixed independently of uniform A , with the same column bound (Theorem 3 and table 3). Chailloux and Hermouet’s one-bit recursion solves binary ISIS using $\Theta_d(N^d)$ columns [6, Section 7.1]. For LtHash, our ISIS algorithm lets an attacker extend two given accumulator states to a collision by solving ISIS with their difference as the target.

Table 3. Polynomial-time column bounds for fixed $q = 2^d$ and uniform A .

Method	Problem	Coefficients	Columns	Success
Regev–CLZ [8, Appendix A]	SIS	$\{0, 1\}$	$(N + 1)^d$	1 (all A)
Chailloux–Hermouet [6]	ISIS	$\{0, 1\}$	$\Theta_d(N^d)$	$1 - \text{negl}(N)$
Kothari et al. [14]	SIS, $q = 4$	$\{-1, 0, 1\}$	$(N + 2)^2$	1 (all A)
This work	SIS and ISIS	$\{-1, 0, 1\}$	$O_d(N^{\lceil d/2 \rceil})$	$\geq 3/4$

Technique. Write $R_s := \mathbb{Z}/2^s\mathbb{Z}$. Our construction replaces the one-bit transition $R_s^N \rightarrow R_{s-1}^N$ by a two-bit transition $R_s^N \rightarrow R_{s-2}^N$.

The layer maintains a basis modulo two. For each trial batch, it selects basis columns making the sum even, then solves a second binary system to choose signs making the sum divisible by four. If the sign system is inconsistent, the layer keeps the basis and tries a new batch. With $t = N + 1$ trial columns, each two-bit layer consumes $O(N)$ inputs in expectation. The recursion uses $\lfloor d/2 \rfloor$ such layers, followed by an extra one-bit layer when d is odd, yielding the $O_d(N^{\lceil d/2 \rceil})$ bound.

Organization. After relating LtHash collisions to ternary SIS and establishing the Regev–CLZ baseline in Section 2, we develop the two-bit recursion and its composition invariants in Section 3, then derive the main column bound in Section 4. Section 5 extends the construction to ISIS. Finally, we give a concrete

cost analysis of LtHash bounds based on our lifting techniques and compare them with prior collision attacks in Section 6.

2 LtHash Collisions and Ternary SIS

2.1 Power-of-Two LtHash and Ternary SIS

Bellare and Micciancio’s original LtHASH adapts Ajtai’s binary-input matrix hash to indexed block sequences over $(\mathbb{Z}/p\mathbb{Z})^k$ [2, Sections 3.1 and 6.1–6.3]. We use the finite-set collision experiment of Lewi et al. and its power-of-two group $(\mathbb{Z}/2^d\mathbb{Z})^N$ [16]. Fix integers $N, d \geq 1$ and an infinite item space $\mathcal{X}$. For $s \geq 1$, put $R_s = \mathbb{Z}/2^s\mathbb{Z}$, and set $R_0 = \{0\}$. In the random oracle model, $H : \mathcal{X} \rightarrow R_d^N$ assigns independent uniform vectors to distinct items.

The domain consists of finite subsets of $\mathcal{X}$, rather than multisets. For such a set S , define

$$\text{LtHash}_{N,d}(S) = \sum_{x \in S} H(x) \in R_d^N,$$

where the empty sum gives $\text{LtHash}_{N,d}(\emptyset) = 0$.

Collision Experiment. Fix $Q \geq 1$. A Q -query adversary $\mathcal{A}$ adaptively queries H on at most Q distinct items and returns finite sets $S, T \subseteq \mathcal{X}$, possibly containing unqueried items. It wins if $S \neq T$ and $\text{LtHash}_{N,d}(S) = \text{LtHash}_{N,d}(T)$. Its success probability is over H and the coins of $\mathcal{A}$.

Definition 1 (SIS and Inhomogeneous SIS). Fix integers $N, m, \beta \geq 1$ and $q \geq 2$. For $A \xleftarrow{\$} (\mathbb{Z}/q\mathbb{Z})^{N \times m}$, the two tasks seek $z \in \mathbb{Z}^m$ with $\|z\|_\infty \leq \beta$ satisfying

$$\begin{aligned} \text{SIS}_\infty(N, m, q, \beta) : \quad & Az = 0 \pmod{q}, \quad z \neq 0, \\ \text{ISIS}_\infty(N, m, q, \beta; b) : \quad & Az = b \pmod{q}. \end{aligned}$$

A solver receives A for SIS and (A, b) for ISIS. Success is measured over A and the solver’s coins. The ISIS guarantee must hold for every $b \in (\mathbb{Z}/q\mathbb{Z})^N$ fixed independently of A .

In Gentry, Peikert, and Vaikuntanathan’s average-case formulation, b is sampled independently and uniformly [11, Definition 5.6].

For $\beta = 1$, we call these problems *ternary SIS* and *ternary ISIS*, with coefficients in $\{-1, 0, 1\}$. Binary solutions use coefficients in $\{0, 1\}$ and also satisfy this coefficient bound.

Collisions as Ternary Relations. The oracle answers to distinct items form independent uniform matrix columns. For distinct queried items $x_1, \dots, x_{Q'}$, place their oracle answers in the columns of A . Every nonzero ternary relation $Az = 0$ gives a collision: take the items indexed by its positive and negative coefficients as the two sets.

Conversely, for a collision between subsets S, T of these queried items, set $z_i = \mathbb{I}[x_i \in S] - \mathbb{I}[x_i \in T]$, where $\mathbb{I}[\cdot]$ is the event indicator. Then $z \neq 0$ is ternary and $Az = 0$. The following reduction also covers unqueried items.

Proposition 1 (Collision Reduction for Power-of-Two LtHash). *If a Q -query adversary $\mathcal{A}$ wins with probability δ , it gives a solver for $\text{SIS}_\infty(N, Q, 2^d, 1)$ that succeeds with probability at least $\delta - 2^{-Nd}$.*

Proof. To simulate a Q -query execution, sample $A = (a_1 \mid \cdots \mid a_Q) \xleftarrow{\$} R_d^{N \times Q}$ and answer the i -th distinct query with a_i , reusing answers on repetitions. The simulation allows adaptive queries and early stopping. Pad relations using fewer than Q columns with zeros. For a returned collision, extract z whenever every item in $S \Delta T$ has been queried. If this symmetric difference contains an unqueried item, choose one by a fixed rule. Conditional on the transcript and all other oracle values, its value is uniform in R_d^N and its coefficient is ± 1 . The collision equation therefore holds with probability 2^{-Nd} . This bounds the lost success probability, as in Lewi et al. [16, Lemma A.2]. $\square$

2.2 Recursion over the Modulus

Recursion over the modulus appears in the SIS self-reduction of Micciancio and Peikert [21] and in the algorithms of G6ös et al. [12]. Chen, Liu, and Zhandry (CLZ) give the column and coefficient bounds used below and attribute the algorithm to Regev (personal communication) [8, Appendix A, Theorems 48–49]. For a power-of-two modulus, any $N + 1$ columns admit a nonzero binary dependence modulo two. The corresponding sum is even, and division by two removes one modulus bit. Each removed bit therefore costs a factor of $N + 1$ in the number of columns. More generally, the recursion follows a factorization of the modulus: each level removes one factor, while the column counts and coefficient bounds multiply.

For $p \mid q$, multiplication by p induces a bijection of additive groups

$$\mathbb{Z}/(q/p)\mathbb{Z} \longrightarrow p(\mathbb{Z}/q\mathbb{Z}), \quad u \longmapsto pu.$$

Division by p denotes its inverse, applied entrywise.

Proposition 2 (Regev–CLZ Recursion). *Fix $N \geq 1$ and a given factorization $q = \prod_{i=1}^k p_i$, where $k \geq 2$ and the integers $p_i \geq 2$ may repeat. Put*

$$Q = (N + 1)^k, \quad \beta = \prod_{i=1}^k \lfloor p_i/2 \rfloor.$$

Given any $A \in (\mathbb{Z}/q\mathbb{Z})^{N \times Q}$, a deterministic classical algorithm returns a nonzero $z \in \mathbb{Z}^Q$ with $Az = 0 \pmod{q}$ and $\|z\|_\infty \leq \beta$, in time polynomial in Q and $\log q$.

Proof. Let $q_j = q / \prod_{i=1}^j p_i$, so $q_0 = q$ and $q_k = 1$. To recover a relation among the original columns, we record how each current column was formed. At level j , each column $v \in (\mathbb{Z}/q_j\mathbb{Z})^N$ carries a nonzero integer representation $\text{rep}(v) \in \mathbb{Z}^Q$ with

$$A \text{rep}(v) = \frac{q}{q_j} v \pmod{q}, \quad \|\text{rep}(v)\|_\infty \leq \prod_{i=1}^j \lfloor p_i/2 \rfloor. \quad (1)$$

The representations have disjoint supports. Initially, $v = a_i$ and $\text{rep}(v) = e_i$.

Partition the current columns into blocks $(v_1, \dots, v_{N+1})$. Linear algebra modulo p_{j+1} , with deterministic choices, gives a nonzero dependence. Take centered integer representatives $x \in \mathbb{Z}^{N+1}$, so that $\sum_i x_i v_i = 0 \pmod{p_{j+1}}$ and $\|x\|_\infty \leq \lfloor p_{j+1}/2 \rfloor$. Replace each block by

$$w = \frac{1}{p_{j+1}} \sum_{i=1}^{N+1} x_i v_i \in (\mathbb{Z}/q_{j+1}\mathbb{Z})^N, \quad \text{rep}(w) = \sum_{i=1}^{N+1} x_i \text{rep}(v_i).$$

Disjoint supports make each nonzero output coefficient a product of one coefficient of x and one input coefficient. Thus $\text{rep}(w)$ remains nonzero, its norm grows by at most a factor of $\lfloor p_{j+1}/2 \rfloor$, and supports remain disjoint between blocks. Substitution in Equation (1) gives $A\text{rep}(w) = (q/q_{j+1})w \pmod{q}$.

Each level divides the number of columns by $N+1$. After k levels, one column over the trivial group remains, and its representation is the required relation. Fewer than Q local eliminations and polynomial-size representation updates give the stated running time. $\square$

For LtHash, take $p_1 = \dots = p_d = 2$ and choose each local dependence with coefficients in $\{0, 1\}$. Disjoint supports preserve these coefficients, giving a nonzero binary relation from $(N+1)^d$ columns. The case $d = 1$ follows from a single linear dependence modulo two. We call this application the *one-bit recursion*.

2.3 Column Bounds and SIS Conjectures

Conjectures for Fixed Moduli. Liyan Chen et al. propose collision-resistant matrix hashes based on ternary SIS [7, full version, Appendix B]. We consider the uniform average-case versions of their conjectures, motivated by this application. A solver must return a valid relation with nonnegligible probability over uniform A and its own randomness.

In this formulation, their Conjecture B.2 states that, for every fixed $d \geq 2$ and constant $\varepsilon > 0$, no $\text{poly}(N)$ -time algorithm solves $\text{SIS}_\infty(N, m, 2^d, 1)$ when $m \leq N^{d-\varepsilon}$. Their Conjecture B.4 asserts, for every fixed $q \geq 4$ and $0 < \varepsilon < 0.99$, that $\text{SIS}_\infty(N, m, q, 1)$ is hard when $m \in O(N^{1+\varepsilon})$. The Regev-CLZ recursion reaches neither regime: it uses $\Theta_d(N^d)$ columns, including $(N+1)^2 = \Theta(N^2)$ at $q = 4$.

Existence Bounds and Algorithms. For $q = 4$, classical zero-sum theory guarantees a nonzero binary relation among any $3N+1$ columns and a nonzero ternary relation among any $2N+1$ columns. These thresholds are the Davenport and plus-minus Davenport constants of $(\mathbb{Z}/4\mathbb{Z})^N$, respectively [18, 26]. For uniform A , each fixed nonzero ternary z makes Az uniform, since some coordinate of z is ± 1 . A union bound therefore gives

$$\Pr[\exists z \in \{-1, 0, 1\}^m \setminus \{0\} : Az = 0] \leq (3^m - 1)4^{-N}.$$

Thus nonnegligible success requires $\Omega(N)$ columns. The algorithmic question is how many columns suffice to find such a relation in polynomial time.

At modulus four, Kothari, O'Donnell, and Wu's ring extension solves SIS with coefficient bound one using $(N + 2)^2$ columns [14, Section 5.1].

Imran and Ivanyos give a deterministic algorithm for signed zero sums over $\mathbb{F}_p^N$, where p is prime. It uses $p^{O(p \log p)} N^{O(p)}$ input vectors and runs in time polynomial in this bound [13].

For prime $q \geq 5$ and $2 \leq k \leq \lfloor q/2 \rfloor$, Kothari, O'Donnell, and Wu give a deterministic SIS algorithm with coefficient bound $\lfloor q/(2k) \rfloor$ whenever

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

The guarantee holds for every input, and the running time is polynomial in $m, k, \log q$ [14, Theorem 4.11]. Their Section 5.1 extends the underlying algorithms to $\mathbb{Z}/q\mathbb{Z}$, with $k = 2$ giving the modulus-four bound above.

Table 3 compares the column counts of polynomial-time algorithms at fixed power-of-two moduli.

Removing Two Bits without Increasing the Coefficient Bound. Using a factor 4 in CLZ's explicit construction removes two bits at once, but gives coefficient bound 2. To obtain a ternary relation with fewer columns, we need a modulus-four step whose coefficient bound is still one. The next section gives such a step using two-bit lifting.

3 Two-Bit Lifting

A one-bit Regev–CLZ layer (Proposition 2) forms a combination in $2R_s^N$ and divides by two. A basis modulo two lets us cancel the parity of a trial batch by adding a unique subset of basis columns. Flipping a selected coefficient from +1 to −1 subtracts twice that column. This preserves the first cancellation but can change the parity after division by two. We then solve a second binary linear system to choose signs that put the combination in $4R_s^N$, so division by four removes two modulus bits. If this system is inconsistent, we retain the basis and try a new batch. Disjoint input supports keep the composed coefficients ternary.

The construction below makes these retries compatible with recursive composition. Section 4 bounds their cost.

3.1 A Two-Bit Lifting Layer

For $r \geq 1$, write $[r] = \{1, \dots, r\}$ and let $\mathbf{1}_r$ be the all-ones vector of length r . We write $\text{supp}(x) = \{i : x_i \neq 0\}$ for the support of a vector or sequence x . If $U = (u_1 \mid \dots \mid u_r)$ and $X \subseteq [r]$, then U_X consists of the columns of U indexed by X , in their original order. For vectors and matrices over R_s , $s \geq 1$, write $\bar{v} = v \bmod 2$ and $\bar{A} = A \bmod 2$ for entrywise reduction to $\mathbb{F}_2$. In integer or ring expressions, we represent each binary coordinate by 0 or 1.

The following lemma expresses these two cancellations as binary linear systems.

Lemma 1 (Two-Bit Lifting). Fix $N \geq 1$, $s \geq 2$, and $t \geq 1$. Let $B = (b_1 \mid \dots \mid b_N) \in R_s^{N \times N}$ have invertible reduction $\bar{B} \in \mathbb{F}_2^{N \times N}$, and let $C = (c_1 \mid \dots \mid c_t) \in R_s^{N \times t}$. Define

$$x = \bar{B}^{-1} \bar{C} \mathbf{1}_t, \quad X = \text{supp}(x), \quad V = (B_X \mid C), \quad u = V \mathbf{1}_{|X|+t}.$$

Then X is the unique subset of $[N]$ satisfying

$$\sum_{i \in X} \bar{b}_i + \sum_{j=1}^t \bar{c}_j = 0. \quad (2)$$

Consequently, $u \in 2R_s^N$. For every $y \in \mathbb{F}_2^{|X|+t}$,

$$V(\mathbf{1}_{|X|+t} - 2y) \in 4R_s^N \iff \bar{V}y = \overline{u/2}. \quad (3)$$

Thus every solution to the binary system on the right gives a lift whose coefficients on V lie in $\{-1, 1\}$.

Proof. Invertibility of $\bar{B}$ makes x the unique solution of $\bar{B}x = \bar{C} \mathbf{1}_t$, proving Equation (2) and $\bar{u} = 0$. Since $V(\mathbf{1}_{|X|+t} - 2y) = u - 2Vy$, dividing by two and reducing modulo two gives Equation (3). $\square$

We implement the lemma using a column source $\mathcal{S}$ and a retained ordered tuple P . Each request returns a new indexed column, even if its value equals an earlier one. We identify columns with their values when unambiguous and write $\text{next}(\mathcal{S})$ for its next column and $\text{next}^r(\mathcal{S})$ for the tuple returned by r consecutive requests.

Auxiliary Operations. We write $P \parallel a$ for appending a to P . The operation $\text{drop}(U)$ discards a column or tuple of columns without changing P or passing the discarded columns to the next layer. If $U\eta \in 4R_s^N$ for $\eta \in \{-1, 1\}^{|U|}$, then $\text{Merge}_4(U, \eta)$ consumes the columns in U and returns a new indexed column with value $U\eta/4 \in R_{s-2}^N$.

For $A \in \mathbb{F}_2^{N \times r}$ and $b \in \mathbb{F}_2^N$, let $\text{Solve}_2(A, b)$ return $\perp$ if $Ay = b$ is inconsistent and otherwise return the solution obtained by Gaussian elimination with a deterministic pivoting rule and all free variables set to zero.

Definition 2 (Two-Bit Lifting Source). For $N \geq 1$, $s \geq 2$, $t \geq 1$, and a column source $\mathcal{S}$ over R_s^N , define $\mathcal{G} = \text{TwoBitLayer}_{s,t}(\mathcal{S})$ as the source over R_{s-2}^N with retained tuple $P = \emptyset$ initially. Each call to $\text{next}(\mathcal{G})$ executes Algorithm 1, preserving the updated P and input-source position between calls. Creating $\mathcal{G}$ makes no input requests. Its execution is deterministic once the input stream is fixed.

A rejection discards only the new trial batch. The basis is changed only after an accepted merge, when the selected basis columns are consumed and the unselected columns are retained.

Algorithm 1 One output request to $\text{TwoBitLayer}_{s,t}(\mathcal{S})$

Require: $N \geq 1$, $s \geq 2$, $t \geq 1$, and a source $\mathcal{S}$ of indexed columns in R_s^N
 $\triangleright$ *Persistent state: an ordered tuple P , initially empty, whose reduced columns are independent*

Ensure: The next indexed output $w \in R_{s-2}^N$, with $\mathcal{S}$ advanced and P updated
 $\triangleright$ *Fill a basis modulo two*

```

1: while  $\text{rk}(\bar{P}) < N$  do
2:    $a \leftarrow \text{next}(\mathcal{S})$ 
3:   if  $\bar{a} \notin \text{span}(\bar{P})$  then
4:      $P \leftarrow P \parallel a$ 
5:   else
6:      $\text{drop}(a)$ 
7:   end if
8: end while
9:  $B \leftarrow P$ 
 $\triangleright$  Try new batches against the basis
10: repeat
11:    $C \leftarrow \text{next}^t(\mathcal{S})$ 
12:    $x \leftarrow \bar{B}^{-1}\bar{C}\mathbf{1}_t$ ,  $X \leftarrow \text{supp}(x)$   $\triangleright$  Cancel the first bit by selecting basis columns
13:    $V \leftarrow (B_X \mid C)$ ,  $u \leftarrow V\mathbf{1}_{|X|+t}$ 
14:    $y \leftarrow \text{Solve}_2(\bar{V}, u/2)$   $\triangleright$  Cancel the second bit by choosing signs
15:   if  $y = \perp$  then
16:      $\text{drop}(C)$ 
17:   end if
18: until  $y \neq \perp$ 
 $\triangleright$  Merge the accepted batch
19:  $P \leftarrow B_{[N] \setminus X}$ 
20:  $\eta \leftarrow \mathbf{1}_{|X|+t} - 2y$ 
21: return  $\text{Merge}_4(V, \eta)$ 

```

3.2 Recursive Composition

As in the one-bit recursion, we track each intermediate column as a signed combination of the original inputs. The base source $\mathcal{G}_0$ returns independent uniform columns $a_1, a_2, \dots \in R_d^N$, with unit-vector representations $\text{rep}(a_i) = e_i$. Write $\{-1, 0, 1\}^{(\mathbb{N})}$ for the finitely supported ternary sequences. A level- j column $v \in R_{d-2j}^N$ carries a nonzero representation $\text{rep}(v) \in \{-1, 0, 1\}^{(\mathbb{N})}$ satisfying

$$4^j v = \sum_{i \geq 1} \text{rep}(v)_i a_i \pmod{2^d}. \quad (4)$$

Its *base support* is $\text{bsupp}(v) = \text{supp}(\text{rep}(v)) \subseteq \mathbb{N}$. To track representations through Merge_4 , write $w = \text{Merge}_4(V, \eta)$ and $V = (v_1 \mid \dots \mid v_r)$. If its inputs have pairwise disjoint base supports, then w carries

$$\text{rep}(w) = \sum_{k=1}^r \eta_k \text{rep}(v_k), \quad \text{bsupp}(w) = \biguplus_{k=1}^r \text{bsupp}(v_k).$$

Each base index contributes through at most one input. The merge transfers the input supports to the output, keeping every nonzero coefficient in $\{-1, 1\}$. We store these representations as signed derivation records and traverse them to recover the terminal relation.

The Recursive Algorithm and Its Terminal Step. Put $L_2 = \lfloor d/2 \rfloor$ and choose batch sizes $t_1, \dots, t_{L_2}$. After L_2 layers, one output gives a relation when d is even, since the residual ring is R_0 . For odd d , a binary dependence among $N+1$ outputs over R_1 cancels the remaining bit.

For $A \in \mathbb{F}_2^{N \times (N+1)}$, define $\text{Kernel}_2(A)$ to return a nonzero vector in $\ker(A)$. It uses the deterministic pivoting rule of Solve_2 , sets the first free variable to one and all others to zero, and solves for the pivot variables. Algorithm 2 composes the sources of Definition 2 and extracts the terminal relation.

Algorithm 2 Recursive ternary relation search by two-bit lifting

Require: $N, d \geq 1$, batch sizes $t_j \geq 1$ for $1 \leq j \leq \lfloor d/2 \rfloor$, and the represented base source $\mathcal{G}_0$

Ensure: A nonzero finitely supported $z \in \{-1, 0, 1\}^{(\mathbb{N})}$ such that $\sum_{i \geq 1} z_i a_i = 0 \pmod{2^d}$

```

1:  $L_2 \leftarrow \lfloor d/2 \rfloor$ 
    $\triangleright$  Build the recursive source
2: for  $j = 1, \dots, L_2$  do
3:    $\mathcal{G}_j \leftarrow \text{TwoBitLayer}_{d-2(j-1), t_j}(\mathcal{G}_{j-1})$ 
4: end for
    $\triangleright$  Extract a terminal relation
5: if  $d$  is even then
6:   return  $\text{rep}(\text{next}(\mathcal{G}_{L_2}))$ 
7: end if
8:  $(w_1, \dots, w_{N+1}) \leftarrow \text{next}^{N+1}(\mathcal{G}_{L_2})$ 
9:  $\lambda \leftarrow \text{Kernel}_2(\bar{w}_1 \mid \dots \mid \bar{w}_{N+1})$ 
10: return  $\sum_{i=1}^{N+1} \lambda_i \text{rep}(w_i)$ 
```

Source Transcripts and Freshness. Downstream layers use the values already returned to decide how many columns to request. The outputs must therefore remain independent and uniform conditional on the internal transcript defined below. In a lifting layer, decisions inspect only the low two bits. One high part from each accepted trial batch supplies an independent mask. Its column is consumed without entering the layer's retained tuple.

For the analysis, fix a stateful source over R_s^N and a value-independent request schedule. Let $\mathbf{T}_m$ be the source's internal transcript through outputs $W_1, \dots, W_m$. It records inspected low residues at each internal modulus, control decisions, costs, base supports, and residual state. The residual state includes the control state and retained indexed columns with their full values and base

supports. The transcript preserves all earlier records and the retained state after every output, so T_{m-1} is determined by T_m . It excludes returned values, downstream observations, and latent future randomness.

Definition 3 (Fresh and Base-Support-Disjoint Sources). *In this model, the source is fresh if, for every deterministic $m \geq 1$ and every attainable transcript t ,*

$$(W_1, \dots, W_m) \mid (\mathsf{T}_m = \mathsf{t}) \sim \text{Unif}((R_s^N)^m). \quad (5)$$

It is base-support-disjoint if, after every finite returned prefix, the base supports of all returned columns and all indexed columns in its residual state are nonempty and pairwise disjoint.

Lemma 2 (High-Part Masks at a Low-Bit Stopping Time). *Let $Z_1, Z_2, \dots$ be returned by deterministic requests to a fresh source over R_s^N , where $s \geq 2$, and write uniquely $Z_i = L_i + 4M_i$ with $L_i \in \{0, 1, 2, 3\}^N$ and $M_i \in R_{s-2}^N$. Put $\mathcal{F}_r^{\text{low}} = \sigma(\mathsf{T}_r, L_1, \dots, L_r)$. The cumulative transcript makes $(\mathcal{F}_r^{\text{low}})$ a filtration. Let $T < \infty$ almost surely be a stopping time for this filtration. For fixed $m \geq 1$, if distinct indices $J_1, \dots, J_m \leq T$ are $\mathcal{F}_T^{\text{low}}$ -measurable, then, conditional on $(T, \mathsf{T}_T, L_1, \dots, L_T)$ and all high parts through time T other than $M_{J_1}, \dots, M_{J_m}$, the selected high parts are independent and uniform in $(R_{s-2}^N)^m$.*

Proof. Equation (5) and the unique decomposition make $(M_1, \dots, M_r)$ uniform conditional on $\mathcal{F}_r^{\text{low}}$ for every $r \geq 1$. For distinct $\mathbf{j} = (j_1, \dots, j_m) \in [r]^m$, the event $\{T = r, (J_1, \dots, J_m) = \mathbf{j}\}$ belongs to this sigma-algebra. Conditioning on it and revealing the other high parts leaves $(M_{j_1}, \dots, M_{j_m})$ independent and uniform. Applying this to every attainable conditioning tuple proves the claim. $\square$

Proposition 3 (Two-Bit Layer Invariant). *Fix $N \geq 1$, $s \geq 2$, and $t \geq 1$. If $\mathcal{S}$ is a fresh, base-support-disjoint source over R_s^N whose calls terminate almost surely, then $\text{TwoBitLayer}_{s,t}(\mathcal{S})$ is a fresh, base-support-disjoint source over R_{s-2}^N , and every output request terminates almost surely.*

Proof. At retained rank $r < N$, freshness makes the next parity uniform, so rank increases with probability $1 - 2^{r-N} \geq 1/2$. Refill therefore terminates almost surely. With a full basis, $x = \bar{B}^{-1}\bar{C}\mathbf{1}_t$ is uniform. The event $x = \mathbf{1}_N$, of probability 2^{-N} , selects the entire basis and makes the sign system consistent. Thus retries also terminate almost surely, for every $t \geq 1$.

For freshness, use the decomposition and filtration of Lemma 2. For fixed $m \geq 1$, the input count $\mathcal{N}_m$ at output m is a stopping time: basis completion, parity selection, and the sign system inspect only low residues. Let J_k index the first trial column in the k -th accepted batch. These indices are distinct, bounded by $\mathcal{N}_m$, and $\mathcal{F}_{\mathcal{N}_m}^{\text{low}}$ -measurable. Write $\mathcal{H}$ for this stopped information together with all high parts through $\mathcal{N}_m$ except the selected masks. With $G = R_{s-2}^N$, the lemma gives

$$(M_{J_1}, \dots, M_{J_m}) \mid \mathcal{H} \sim \text{Unif}(G^m).$$

Accepted trial columns are consumed without entering the retained tuple, so no selected mask enters another output's offset or any recorded retained state. Thus each output's sign η_k and offset D_k are fixed by $\mathcal{H}$, and an affine bijection gives

$$W_k = \eta_k M_{J_k} + D_k, \quad \eta_k \in \{-1, 1\}, \quad (6)$$

$$(W_1, \dots, W_m) \mid \mathcal{H} \sim \text{Unif}(G^m).$$

Since $\mathcal{H}$ determines the output source's cumulative transcript, averaging over data with the same transcript proves Equation (5). Disjoint consumption preserves nonempty, pairwise disjoint base supports. $\square$

Theorem 1 (Correctness of the Recursive Search). *Fix $N, d \geq 1$ and batch sizes $t_j \geq 1$ for $1 \leq j \leq \lfloor d/2 \rfloor$. On a stream of independent uniform columns in R_d^N , Algorithm 2 terminates almost surely and returns a nonzero $z \in \{-1, 0, 1\}^{\mathcal{Q}}$ satisfying $\sum_{i=1}^{\mathcal{Q}} z_i a_i = 0 \pmod{2^d}$, where $\mathcal{Q}$ is the number of requests made to the base source before termination.*

Proof. The base source is fresh with disjoint singleton supports. Proposition 3 propagates these properties and almost-sure termination through every layer. Substituting the input representations in the merge rule preserves Equation (4), and disjoint supports keep the resulting representation ternary and nonzero. For even d , the terminal value lies in R_0^N , giving the relation. For odd d , the nonzero terminal binary dependence selects nonempty disjoint supports and cancels the last bit. It therefore also gives a nonzero ternary relation, including when $d = 1$. $\square$

4 Complexity of Two-Bit Lifting

We first count the inputs spent on lifting attempts and basis refill, then compose the expected costs across layers. For constant d , truncating the recursion yields a polynomial-time SIS solver.

4.1 Success Probability and Expected Cost of One Layer

With a retained basis B , one attempt in Algorithm 1 uses a new batch of t columns and succeeds when its sign system is consistent. We take the quotient by the $|X|$ selected basis directions, leaving dimension $N - |X|$. Success then means that a uniform target lies in the span of a uniform binary matrix. For integers $a, b \geq 0$ and independent uniform $G \in \mathbb{F}_2^{a \times b}$, $h \in \mathbb{F}_2^a$, define

$$\gamma(a, b) = \Pr[h \in \text{colspan}(G)] = \mathbb{E}\left[2^{\text{rk}(G) - a}\right]. \quad (7)$$

Lemma 3 (One-Attempt Success Probability). *In the setting of Lemma 1, let $C \stackrel{\$}{\leftarrow} R_s^{N \times t}$, put $K = |X|$, and let success denote consistency of the sign system in Algorithm 1. Then*

$$K \sim \text{Bin}(N, 1/2), \quad \Pr[\text{success} \mid K = k] = \gamma(N - k, t - 1).$$

Thus an attempt succeeds with probability

$$p_{N,t} = 2^{-N} \sum_{k=0}^N \binom{N}{k} \gamma(N-k, t-1). \quad (8)$$

The joint law of $(K, \text{success})$ is independent of the basis B .

Proof. Put $D = \bar{B}^{-1}\bar{C} = (d_1, \dots, d_t)$. Its columns are uniform and independent, so $x = D\mathbf{1}_t$ is uniform and K is binomial. Conditional on x , the first $t-1$ columns remain uniform and independent, and $d_t = x + \sum_{j<t} d_j$. In the quotient by $E_X = \langle e_i : i \in X \rangle$, their span is therefore generated by a uniform matrix $G \in \mathbb{F}_2^{(N-K) \times (t-1)}$, since $x \in E_X$.

Conditional on the parities, write $c_j = \delta_j + 2h_j \pmod{4}$, where $\delta_j \in \{0, 1\}^N$ represents $\bar{c}_j$ and the $h_j \in \mathbb{F}_2^N$ are independent and uniform. For κ fixed by $B \pmod{4}$ and these parities,

$$\bar{B}^{-1}u/2 = \kappa + \bar{B}^{-1} \sum_{j=1}^t h_j.$$

As $t \geq 1$, the quotient target is uniform independently of G . Thus Equation (7) gives the conditional success law, and averaging over K gives Equation (8). The binomial law and these conditional probabilities do not depend on B . $\square$

When the retained columns have rank $N-j$, a fresh parity increases the rank with probability $1 - 2^{-j}$. Thus, after consuming k basis columns, the expected number of requests needed to restore the basis is

$$r_k = \sum_{j=1}^k \frac{1}{1 - 2^{-j}}, \quad r_0 = 0. \quad (9)$$

Acceptance biases the law of K , so the relevant refill cost is

$$R_{N,t} = \mathbb{E}[r_K \mid \text{success}] = \frac{2^{-N}}{p_{N,t}} \sum_{k=0}^N \binom{N}{k} \gamma(N-k, t-1) r_k. \quad (10)$$

Proposition 4 (Expected Input Count for One Layer). *Let the input be a fresh source whose calls terminate almost surely. Starting with $P = \emptyset$, the expected number of input-source requests required to return $m \geq 1$ outputs is*

$$F_{N,t}(m) = r_N + \frac{mt}{p_{N,t}} + (m-1)R_{N,t}. \quad (11)$$

Proof. Filling the initial basis costs r_N expected requests. By Lemma 3, every retained basis gives a geometric number of attempts of mean $1/p_{N,t}$, each using t inputs. Each of the first $m-1$ accepted outputs also incurs refill cost $R_{N,t}$, and there is no refill after the last output. Summing gives Equation (11). $\square$

The finite binary rank distribution used to evaluate γ is recorded in Appendix A.

4.2 Expected Cost of the Recursion

For real $x \geq 1$, extend the cost in Equation (11) as

$$\tilde{F}_{N,t}(x) = \mu_{N,t}x + c_{N,t}, \quad (12)$$

$$\mu_{N,t} = \frac{t}{p_{N,t}} + R_{N,t}, \quad c_{N,t} = r_N - R_{N,t}. \quad (13)$$

The coefficient $\mu_{N,t}$ is the expected cost per additional output, and $c_{N,t}$ accounts for the initial basis and the omitted final refill. An upper layer uses the returned values to decide how many columns to request. The next lemma shows that we can apply the same affine map to the expected demand.

Lemma 4 (Affine Cost at an Output Stopping Time). *Let $W_1, W_2, \dots$ be the values returned by a fresh source, and let $\mathcal{C}(m) \geq 0$ be the cost of producing the first m outputs, determined by the source transcript through output m . Suppose that, for every integer $m \geq 1$, $\mathbb{E}[\mathcal{C}(m)] = \alpha m + \beta < \infty$. Let ρ be auxiliary randomness independent of the source, and let $\mathcal{D} \geq 1$ be an almost-surely finite stopping time for $\mathcal{F}_m^{\text{out}} = \sigma(W_1, \dots, W_m, \rho)$, with $\mathbb{E}[\mathcal{D}] < \infty$. Then*

$$\mathbb{E}[\mathcal{C}(\mathcal{D})] = \alpha \mathbb{E}[\mathcal{D}] + \beta. \quad (14)$$

If $\mathbb{E}[\mathcal{C}(m)] \leq \alpha m + \beta$ for every $m \geq 1$, the conclusion holds with $\leq$.

Proof. For each m , freshness and independence of ρ make $\mathcal{C}(m)$ independent of $\mathcal{F}_m^{\text{out}}$. Since $\{\mathcal{D} = m\} \in \mathcal{F}_m^{\text{out}}$, partitioning by $\mathcal{D}$ and using nonnegativity gives

$$\mathbb{E}[\mathcal{C}(\mathcal{D})] = \sum_{m \geq 1} \mathbb{E}[\mathcal{C}(m)] \Pr[\mathcal{D} = m] = \alpha \mathbb{E}[\mathcal{D}] + \beta.$$

Substituting upper bounds for the expectations at each fixed output count proves the inequality version. $\square$

Proposition 5 (Expected Number of Base Columns). *Fix $N, d \geq 1$, put $L_2 = \lfloor d/2 \rfloor$, and choose batch sizes $t_j \geq 1$ for $1 \leq j \leq L_2$. Let the base source return independent uniform columns in R_d^N , and let $\mathcal{Q}$ be the total number of requests made to it by Algorithm 2. Put $m_\star = 1$ for even d and $m_\star = N + 1$ for odd d . Then*

$$\mathbb{E}[\mathcal{Q}] = \tilde{F}_{N,t_1} \left(\tilde{F}_{N,t_2} \left(\dots \tilde{F}_{N,t_{L_2}}(m_\star) \dots \right) \right), \quad (15)$$

where the empty composition is the identity.

Proof. By Proposition 3, every layer is fresh and terminates almost surely on each finite requested prefix. Starting with the deterministic terminal demand $m_\star$, work downward: demand from above is a stopping time for the returned values, while the input cost is measured by the source's local transcript. At each layer, Lemma 4 applies $\tilde{F}_{N,t_j}$ to the expected demand. Induction gives finite means at every layer and the composition in Equation (15). $\square$

For a common batch size t , let Q_ℓ be the expected input count for $m_\star$ outputs through ℓ layers. The recurrence $Q_0 = m_\star$, $Q_{\ell+1} = \mu_{N,t}Q_\ell + c_{N,t}$ gives

$$Q_\ell = \mu_{N,t}^\ell m_\star + c_{N,t} \frac{\mu_{N,t}^\ell - 1}{\mu_{N,t} - 1}, \quad \mathbb{E}[\mathcal{Q}] = Q_{L_2}. \quad (16)$$

4.3 Ternary SIS for Constant d

For fixed d , take $t = N + 1$ at every layer. The resulting expected input cost per output is linear in N , replacing the quadratic factor $(N + 1)^2$ of two successive one-bit Regev–CLZ levels. For $0 \leq a \leq N$, the full-row-rank event gives

$$\gamma(a, N) \geq \prod_{j=N-a+1}^N (1 - 2^{-j}) \geq \prod_{j=1}^N (1 - 2^{-j}) > \frac{1}{4}.$$

Indeed, separate the factor $1/2$ and use $\prod_{j=2}^N (1 - 2^{-j}) \geq 1 - \sum_{j=2}^N 2^{-j} > 1/2$. Hence

$$p_{N,N+1} > \frac{1}{4}. \quad (17)$$

Also, $r_N < N + 2$ and $0 \leq R_{N,N+1} \leq r_N$ give $\mu_{N,N+1} < 5N + 6 \leq 11N$ and $0 \leq c_{N,N+1} < N + 2 \leq 3N$.

Theorem 2 (Ternary SIS at a Fixed Power-of-Two Modulus). *Fix $d \geq 1$, put $L_2 = \lfloor d/2 \rfloor$, and set*

$$U_d(N) = 2 \cdot 14^{L_2} N^{\lceil d/2 \rceil}, \quad M_d(N) = \lceil 4U_d(N) \rceil. \quad (18)$$

There is a classical algorithm, uniform in N , such that for $A \xleftarrow{\$} R_d^{N \times M_d(N)}$, every execution runs in $\text{poly}(N)$ time and, with probability at least $3/4$, returns a nonzero $z \in \{-1, 0, 1\}^{M_d(N)}$ satisfying $Az = 0 \pmod{2^d}$. In particular, $M_d(N) = O_d(N^{\lceil d/2 \rceil})$.

Proof. Since $Q_\ell \geq 1$, the bounds above give $Q_{\ell+1} < 14NQ_\ell$. We have $Q_0 = 1$ for even d and $Q_0 = N + 1 \leq 2N$ for odd d . Equation (16) therefore yields

$$\mathbb{E}[Q] \leq U_d(N). \quad (19)$$

Feed the columns of A to the base source and abort before requesting column $M_d(N) + 1$. Markov's inequality bounds the abort probability by $1/4$. If the execution does not abort, it returns the required relation by Theorem 1. Pad unused coordinates with zeros.

Across the $O_d(1)$ layers, disjoint consumption and the cap bound the total number of basis-filling iterations and lifting attempts by $O_d(N^{\lceil d/2 \rceil})$. Each such step performs binary elimination or ring arithmetic on data of polynomial size, and representation extraction is polynomial as well. $\square$

Corollary 1 (Refutation of the Average-Case Conjectures). *The uniform average-case formulation of Liyan Chen et al.'s Conjecture B.2 in Section 2.3 is false for every fixed $d \geq 2$. The same formulation of their Conjecture B.4 is false at modulus four [7, full version, Appendix B].*

Proof. Let $a = \lceil d/2 \rceil$. For any fixed $0 < \varepsilon < d - a$, the bound $M_d(N) = O_d(N^a)$ lies below $N^{d-\varepsilon}$ for sufficiently large N , within the range claimed hard in Conjecture B.2. At $d = 2$, the theorem uses $O(N)$ columns and therefore also lies below $N^{1+\varepsilon}$ for any fixed $0 < \varepsilon < 0.99$, contradicting the corresponding average-case claim for uniformly random A in Conjecture B.4. $\square$

Consequences for the Matrix-Hash Proposal. At modulus four, the solver uses only $O(N)$ columns. For a uniformly sampled matrix with a superlinear number of columns, it runs on a sufficiently long prefix and pads the remaining coefficients with zeros. Hence the hash $x \mapsto Ax \bmod 4$ proposed by Liyan Chen et al. [7, Appendix B] is not collision resistant against polynomial-time adversaries.

This rules out the modulus-four candidate as an instantiation of their Theorems 5.2 and 6.1 [7] for any adversary class containing our solver.

For fixed $d \geq 3$, hardness in the smaller superlinear range $m = N^{1+\delta}$, $0 < \delta < \lceil d/2 \rceil - 1$, could still support the constant-depth composition of Liyan Chen et al. [7, Appendix B].

5 Inhomogeneous Extension

The homogeneous result extends to $Az = b$ for every target b fixed independently of the uniform input matrix, with the same asymptotic column bound. For LtHash, this lets us extend two distinct previously queried sets to a collision by canceling their fixed hash difference with fresh columns.

5.1 The Chailloux–Hermouet ISIS Extension

Chailloux and Hermouet extend the Regev–CLZ recursion to ISIS. For $q = 2^d$, their solver uses $m = w^d$ columns, where $w > CN$ for a constant $C > 1$. For every target $b \in R_d^N$ fixed independently of a uniform A , it returns $z \in \{0, 1\}^m$ satisfying $Az = b$, except with negligible probability in N [6, Proposition 10]. Each level removes one modulus bit by choosing between two binary preimages in each block [6, Section 7.1, Algorithm 2].

Taking $w = \Theta(N)$ gives $m = \Theta_d(N^d)$ columns and polynomial running time for fixed d [6, Proposition 11].

5.2 Lifting a Target through the Recursion

We extend the two-bit recursion of Section 3 with a target τ , initially set to $-b$. At each lift, we keep its coefficient at $+1$ and choose signs for the source columns. Its base support is initially empty and records the base columns that contribute to τ .

Lemma 5 (Inhomogeneous Two-Bit Lifting). *Fix $N \geq 1$, $s \geq 2$, and $t \geq 1$. Let $B \in R_s^{N \times N}$ have invertible reduction, let $\tau \in R_s^N$, and let $C \in R_s^{N \times t}$. Set*

$$x = \bar{B}^{-1}(\bar{\tau} + \bar{C}\mathbf{1}_t), \quad X = \text{supp}(x), \quad V = (B_X \mid C), \quad u = \tau + V\mathbf{1}_{|X|+t}.$$

Then $u \in 2R_s^N$, and for every $y \in \mathbb{F}_2^{|X|+t}$,

$$\tau + V(\mathbf{1}_{|X|+t} - 2y) \in 4R_s^N \iff \bar{V}y = \overline{u/2}. \quad (20)$$

A solution produces $\tau' = (\tau + V(\mathbf{1}_{|X|+t} - 2y))/4$. The numerator uses coefficient $+1$ on the incoming target and coefficients in $\{-1, 1\}$ on the source columns.

Proof. We have $\bar{u} = \bar{\tau} + \bar{B}x + \bar{C}\mathbf{1}_t = 0$ and $\tau + V(\mathbf{1}_{|X|+t} - 2y) = u - 2Vy$. Dividing the latter identity by two and reducing modulo two proves the equivalence. $\square$

Definition 4 (Inhomogeneous Lifting Step). For $N \geq 1$, $s \geq 2$, $t \geq 1$, a target $\tau \in R_s^N$, and a column source $\mathcal{S}$ over R_s^N , write $\text{InhomLift}_{s,t}(\tau, \mathcal{S}) \rightarrow (\tau', \mathcal{S}')$ for the following procedure. Build a basis B from an empty tuple using $\mathcal{S}$, as in Algorithm 1. Apply the parity correction of Lemma 5 to new batches of t columns and solve the sign system with Solve_2 , retaining (τ, B) after each rejection. On acceptance, consume the selected basis columns and trial batch, discard the other basis columns, and return the lifted target $\tau' \in R_{s-2}^N$ together with the unused tail $\mathcal{S}'$ of the input source, still over R_s^N .

Before the next target lift, a homogeneous two-bit layer lowers the source modulus to match the target.

Proposition 6 (Inhomogeneous Lifting Invariant). Fix $N \geq 1$, $s \geq 2$, and $t \geq 1$, together with an incoming target $\tau \in R_s^N$ and its base support. In the setting of Lemma 5, with $C \stackrel{\$}{\leftarrow} R_s^{N \times t}$, let $\text{success}_{\text{inh}}$ denote consistency of the sign system, and put $K = |X|$. Then

$$K \sim \text{Bin}(N, 1/2), \quad \Pr[\text{success}_{\text{inh}} \mid K = k] \geq \gamma(N - k, t - 1), \quad (21)$$

where γ is defined in Equation (7). For $t = N + 1$, the attempt success probability exceeds $1/4$ uniformly in B and τ . Moreover, if the incoming source $\mathcal{S}$ is fresh, base-support-disjoint from τ , and its calls terminate almost surely, then an inhomogeneous lift terminates almost surely, the outgoing target τ' is uniform in R_{s-2}^N , and conditional on τ' and its updated base support, the residual source $\mathcal{S}'$ remains fresh and base-support-disjoint.

Proof. Put $a = \bar{B}^{-1}\bar{\tau}$ and $D = \bar{B}^{-1}\bar{C} = (d_1, \dots, d_t)$. Then $x = a + D\mathbf{1}_t$ is uniform in $\mathbb{F}_2^N$, so $K \sim \text{Bin}(N, 1/2)$. Conditional on x , the images of $d_1, \dots, d_{t-1}$ in the quotient by $E_X = \langle e_i : i \in X \rangle$ form a uniform matrix $G \in \mathbb{F}_2^{(N-K) \times (t-1)}$, since $d_t = a + x + \sum_{j < t} d_j$. As in Lemma 3, the next bits of C give an independent uniform target in this quotient. The full quotient span contains $\text{colspan}(G)$, so

$$\Pr[\text{success}_{\text{inh}} \mid x] \geq \mathbb{E}_G \left[2^{\text{rk}(G) - (N-K)} \right] = \gamma(N - K, t - 1).$$

Averaging over x with $|X| = k$ gives Equation (21). For $t = N + 1$, averaging over K gives success probability greater than $1/4$ by Equation (17). For every $t \geq 1$, the event $x = \mathbf{1}_N$ instead gives the bound 2^{-N} , which suffices for almost-sure termination of retries. Basis refill terminates as in Proposition 3.

For joint freshness, write the requested columns as $Z_i = L_i + 4M_i$, and let T count the requests through acceptance. The low-bit decisions make T an almost-surely finite stopping time for $(\mathcal{F}_r^{\text{low}})_r$. Let J index the first column of the accepted batch, with selected sign $\epsilon \in \{-1, 1\}$. Fix $m \geq 0$ and a realization $T = r$, and put $\mathbf{Z} = (Z_{r+1}, \dots, Z_{r+m})$. Write $\mathcal{H}$ for the source transcript through request $r+m$, the low residues through r , and all high parts through r except M_J .

These data determine T, J, ϵ and the updated target support. By the conditioning argument of Lemma 2,

$$(M_J, \mathbf{Z}) \mid \mathcal{H} \sim \text{Unif}(R_{s-2}^N \times (R_s^N)^m).$$

For an offset $D' \in R_{s-2}^N$ fixed by $\mathcal{H}$, an affine bijection gives

$$\begin{aligned} \tau' &= \epsilon M_J + D', \\ (\tau', \mathbf{Z}) \mid \mathcal{H} &\sim \text{Unif}(R_{s-2}^N \times (R_s^N)^m). \end{aligned} \tag{22}$$

Since $\mathcal{H}$ also determines the residual source's transcript, averaging over these data and r proves freshness conditional on τ' and its updated support. Disjoint consumption preserves base-support disjointness. $\square$

The Inhomogeneous Recursion. Fix $b \in R_d^N$, let the base source return independent uniform columns, set $\tau_0 = -b$, and put $L_2 = \lfloor d/2 \rfloor$. At each stage $1 \leq j \leq L_2$, $\text{InhomLift}_{d-2(j-1), t_j}$ from Definition 4 lowers the target modulus by two bits. Whenever another target lift or the odd- d terminal step remains, a homogeneous two-bit layer with the same batch size lowers the residual source modulus to match the new target. The target representations satisfy

$$4^j \tau_j = -b + \sum_{i \geq 1} z_i^{(j)} a_i \pmod{2^d}, \quad z^{(j)} \in \{-1, 0, 1\}^{(\mathbb{N})}. \tag{23}$$

The set $\text{supp}(z^{(j)})$ remains disjoint from the base supports carried by the residual source.

For even d , the final target lies in R_0^N , so the recursion stops without constructing the last homogeneous layer. For odd $d \geq 3$, that layer supplies a source over R_1 , and a basis from this source cancels the target's remaining bit. When $d = 1$, the base source is already over R_1 : use this terminal basis step for $b \neq 0$, and the homogeneous kernel step for $b = 0$.

Proposition 7 (Correctness of the ISIS Recursion). *Fix $N, d \geq 1$, choose $t_j \geq 1$ for $1 \leq j \leq \lfloor d/2 \rfloor$, and fix $b \in R_d^N$. For independent uniform columns $a_1, a_2, \dots \in R_d^N$, the recursion above terminates almost surely and returns a nonzero finitely supported $z \in \{-1, 0, 1\}^{(\mathbb{N})}$ satisfying $\sum_{i \geq 1} z_i a_i = b \pmod{2^d}$.*

Proof. At $j = 0$, $\tau_0 = -b$ has empty base support and the base source is fresh, so Equation (23) holds. At stage j , let V_j be the tuple of consumed source columns and $\eta^{(j)} \in \{-1, 1\}^{|V_j|}$ their signs. Multiply $4\tau_j = \tau_{j-1} + V_j \eta^{(j)}$ by 4^{j-1} and substitute the level- $(j-1)$ target and source representations. The incoming target has coefficient $+1$, and the consumed supports are pairwise disjoint and disjoint from its support. Thus existing coefficients are preserved and new ones lie in $\{-1, 1\}$, proving Equation (23) at level j . By Proposition 6, the residual source is fresh conditional on the updated target and its support. Any intervening homogeneous layer preserves these properties by Proposition 3. This completes the induction.

These propositions also show that every lift terminates almost surely. For odd d , binary rank growth gives the same guarantee for the terminal basis refill.

For even $d = 2L_2 \geq 2$, the final state lies in R_0^N , and Equation (23) gives $Az = b \pmod{2^d}$. For odd $d = 2L_2 + 1 \geq 3$, let $B_\star$ be the terminal basis modulo two and solve $\bar{B}_\star x = \bar{\tau}_{L_2}$. Adding the representations selected by x changes the right-hand side of Equation (23) by $2^{d-1}\tau_{L_2}$. The two copies of this last-bit residue cancel modulo 2^d , again giving $Az = b$. Disjoint base supports preserve ternary coefficients.

For $d \geq 2$, the first accepted lift consumes a nonempty trial batch. Its source columns have nonempty supports, and the induction preserves their nonzero coefficients, so $z \neq 0$ in both terminal cases. For $d = 1$ and $b \neq 0$, the terminal-basis solve selects a nonempty set and returns $Az = b$. For $b = 0$, the homogeneous kernel step on $N + 1$ columns returns a nonzero binary relation. $\square$

5.3 Column Bound and Accumulator Extensions

For ternary ISIS, put $L_2 = \lfloor d/2 \rfloor$ and define

$$U_d^{\text{tar}}(N) = (L_2 + 1)(11N)^{\lceil d/2 \rceil}, \quad M_d^{\text{tar}}(N) = \lceil 4U_d^{\text{tar}}(N) \rceil. \quad (24)$$

Theorem 3 (Ternary ISIS at a Fixed Power-of-Two Modulus). *Fix $d \geq 1$. There is a deterministic classical algorithm $\mathcal{A}_d$, uniform in N , such that for every target $b \in R_d^N$ and independently sampled $A \xleftarrow{\$} R_d^{N \times M_d^{\text{tar}}(N)}$, every execution runs in $\text{poly}(N)$ time and, with probability at least $3/4$, returns a nonzero $z \in \{-1, 0, 1\}^{M_d^{\text{tar}}(N)}$ satisfying $Az = b \pmod{2^d}$. Thus ternary ISIS is solvable with $O_d(N^{\lceil d/2 \rceil})$ columns.*

Proof. Run the inhomogeneous recursion with $t = N + 1$. By Proposition 6 and equation (17), either an inhomogeneous lift or an output of a homogeneous layer uses at most $r_N + 4(N + 1) < 5N + 6 \leq 11N$ expected requests to its input source, uniformly in the target and any retained basis. Conditional on each updated target and its support, we use residual freshness and Lemma 4 to compose this bound through the intervening layers. The stage- j lift therefore costs at most $(11N)^j$ expected base requests. For odd d , the terminal basis costs at most $(11N)^{L_2+1}$, and the $d = 1, b = 0$ kernel step uses only $N + 1 \leq 11N$. Thus the total number $\mathcal{Q}_b$ of base requests satisfies

$$\begin{aligned} \mathbb{E}[\mathcal{Q}_b] &\leq \sum_{j=1}^{\lceil d/2 \rceil} (11N)^j \\ &\leq (L_2 + 1)(11N)^{\lceil d/2 \rceil} = U_d^{\text{tar}}(N). \end{aligned}$$

Define $\mathcal{A}_d$ by aborting before requesting base column $M_d^{\text{tar}}(N) + 1$. Markov's inequality bounds the abort probability by $1/4$. If the execution does not abort, Proposition 7 supplies the required solution, padded with zeros. Counting disjointly consumed supports as in Theorem 2 bounds the number of basis-filling

iterations and lifting attempts by $O_d(N^{\lceil d/2 \rceil})$. Binary elimination, ring arithmetic, and representation extraction are polynomial, giving the running-time bound on every capped execution. $\square$

Taking the fixed difference of two accumulator states as the target gives the following extension guarantee.

Corollary 2 (Extending Accumulator States to a Collision). *Fix $d \geq 1$, condition on an attainable random-oracle transcript, and let P, Q be distinct queried sets determined by it. Suppose $M_d^{\text{tar}}(N)$ distinct, previously unqueried items outside $P \cup Q$ can be selected efficiently. Then a $\text{poly}(N)$ -time algorithm queries these items and, with probability at least $3/4$, returns disjoint sets U, V of newly queried items such that*

$$P \cup U \neq Q \cup V, \quad \text{LtHash}_{N,d}(P \cup U) = \text{LtHash}_{N,d}(Q \cup V). \quad (25)$$

The additional-query count is $O_d(N^{\lceil d/2 \rceil})$.

Proof. Put $b = \text{LtHash}_{N,d}(Q) - \text{LtHash}_{N,d}(P)$. The new answers form a uniform matrix A independent of b . Apply Theorem 3 and split its solution by sign. The hash equality follows from $-b + Az = 0$. Since the added items lie outside $P \cup Q$ and the sign supports are disjoint, $P \neq Q$ makes the final sets distinct. $\square$

6 Concrete Bounds and Applications

For LtHash16, query and scalar-operation budgets of approximately $2^{83.07}$ and $2^{102.95}$, respectively, suffice for collision search with success probability at least $1/2$. We derive these budgets and compare prior attacks at the same parameters, then examine how the costs change with the modulus and message format.

6.1 Concrete Costs for LtHash16

LtHash16 uses $(N, d) = (1024, 16)$ [1,9,15,20,27,28]. We apply the homogeneous recursion of Algorithm 2, with eight two-bit layers and one terminal output. Fix distinct items $x_1, x_2, \dots$, independently of the random oracle, and supply $a_i = H(x_i)$ as the base columns. Each requested base column costs one oracle query. Let $\mathcal{Q}$ denote the total query count in the uncapped execution.

Batch Sizes and Expected Queries. The asymptotic analysis uses $t = N + 1$. Minimizing the exact expected query count for LtHash16 gives $t = 550$ at all eight layers, balancing trial batches, retries, and basis refill. Appendix A.1 proves optimality among fixed positive-integer schedules. We use this schedule throughout the evaluation of our LtHash16 attack.

At $(N, t) = (1024, 550)$, the finite rank sums give

$$p_{1024,550} \approx 0.98982, \quad \tilde{F}_{1024,550}(x) \approx 1069.70x + 511.57.$$

Starting from one terminal output, compute the expected input counts recursively:

$$Q_0 = 1, \quad Q_{j+1} = \tilde{F}_{1024,550}(Q_j) \quad (0 \leq j < 8). \quad (26)$$

By Proposition 5, $\mathbb{E}[\mathcal{Q}] = Q_8$. Evaluation with the full-precision coefficients gives $\log_2 Q_8 \approx 81.07$.

Expected Scalar Work. We count one XOR or AND over $\mathbb{F}_2$ as one scalar operation. Let $\mathcal{L}_\oplus$ and $\mathcal{L}_\otimes$ be their respective counts and put $\mathcal{L}_{\mathbb{F}_2} = \mathcal{L}_\oplus + \mathcal{L}_\otimes$. This model expands ring arithmetic into Boolean circuits and includes terminal sign propagation. The count excludes oracle evaluation, memory traffic, branches, and pointer operations. We count oracle calls separately in $\mathcal{Q}$. Summing the expected costs of lifting attempts, basis factorizations, accepted outputs, and terminal sign propagation gives the bound $\hat{L}_{N,d,t}$ derived in Appendix A.2. For LtHash16,

$$\mathbb{E}[\mathcal{L}_{\mathbb{F}_2}] \leq \hat{L}_{1024,16,550} < 2^{100.95}. \quad (27)$$

Corollary 3 (Concrete Collision Bound for LtHash16). *For the eight-layer LtHash16 attack with batch size 550, put*

$$B_Q = \lceil 4Q_8 \rceil, \quad B_L = \lceil 4\hat{L}_{1024,16,550} \rceil.$$

Abort before the next query or scalar operation would exceed its budget. In the random oracle model, this execution returns two distinct finite sets with the same LtHash_{1024,16} value with probability at least 1/2.

Proof. The independent uniform oracle columns satisfy the hypotheses of Theorem 1. The uncapped execution therefore returns a nonzero finite ternary relation z almost surely. The items indexed by its positive and negative supports form distinct colliding sets S and T , as in Section 2.1.

Run the capped and uncapped algorithms with the same oracle answers. Whenever $\mathcal{Q} \leq B_Q$ and $\mathcal{L}_{\mathbb{F}_2} \leq B_L$, the capped execution completes the same computation and returns the same collision. Using $\mathbb{E}[\mathcal{Q}] = Q_8$ and Equation (27), Markov's inequality and a union bound give

$$\Pr[\mathcal{Q} \leq B_Q, \mathcal{L}_{\mathbb{F}_2} \leq B_L] \geq 1 - \frac{Q_8}{B_Q} - \frac{\hat{L}_{1024,16,550}}{B_L} \geq \frac{1}{2}. \quad (28)$$

□

Memory and Collision Size. Including signed derivation records, expected peak space is below $2^{88.45}$ bits. The query-capped execution uses less than $2^{89.66}$ bits. These bounds, proved in Appendix A.3, exclude memory used internally to evaluate H , external message storage, and the write-only output stream. The fixed buffers for vector arithmetic and linear solves occupy less than 16 MiB. A final traversal of the signed derivation records recovers the collision. In the uncapped execution, the returned sets have total size $|S| + |T| = |\text{supp}(z)| \geq 550^8 \approx 2^{72.83}$, with expectation approximately $2^{80.43}$.

6.2 Comparison with Prior Collision Attacks

At the LtHash16 parameters, two-bit lifting improves both the query estimate and the scalar-work bound of the one-bit recursion.

The CPW and MS rows of Table 2 reproduce the query counts and running-time estimates reported by Lewi et al. [16, Table 1]. We estimate their peak space by evaluating both merge trees in post-order, using TSG’s index trimming for collision recovery [29, Appendix D.1].

TSG’s single-list algorithm applies directly to the values $H(x)$, and its query, time, and space formulas use the bit length $Nd = 16384$ of each group element [29, Theorems 4–5 and Appendix D.2]. Its time estimate uses its list-processing model. The Regev–CLZ recursion uses $1025^{16} \approx 2^{160.02}$ queries. For time and space, we analyze dense block elimination with a depth-first traversal and stored bits identifying the selected columns. Our evaluations of both recursions count scalar operations over $\mathbb{F}_2$. Appendix A.4 derives these evaluations.

6.3 Dependence on LtHash Parameters

For LtHash20 and LtHash32, use $(N, d, t) = (1008, 20, 542)$ and $(1024, 32, 550)$, respectively, with $L_2 = d/2$ homogeneous layers. The expected query count is Q_{L_2} from Equation (16). The expected scalar work is at most $\hat{L}_{N,d,t}$ from Appendix A.2.

Taking $B_Q = \lceil 4Q_{L_2} \rceil$ and $B_L = \lceil 4\hat{L}_{N,d,t} \rceil$, the proof of Corollary 3 gives success probability at least $1/2$ within both budgets. Table 4 evaluates them for all three instances: LtHash20 lies below 128 bits and LtHash32 below 192 bits.

Table 4. Query and scalar $\mathbb{F}_2$ -operation budgets for collision search on power-of-two LtHash, with success probability at least $1/2$.

Name	(N, d)	t	$\log_2 B_Q$	$\log_2 B_L$
LtHash16	(1024, 16)	550	83.07	102.95
LtHash20	(1008, 20)	542	102.97	122.85
LtHash32	(1024, 32)	550	163.57	183.59

Figure 1 places these instances in the larger family with even d . At $N = 1024$ and $t = 550$, both budgets remain below 2^{128} through $d = 20$, below 2^{192} through $d = 32$, and below 2^{256} through $d = 46$. The next even value of d crosses each threshold.

6.4 Indexed Blocks and iSHAKE

Incremental SHAKE [22] (iSHAKE) changes both the modulus exponent and the admissible messages. It instantiates the indexed-block LtHASH in $(\mathbb{Z}/2^{64}\mathbb{Z})^N$. iSHAKE has a different setting from the LtHash scenario discussed above: its

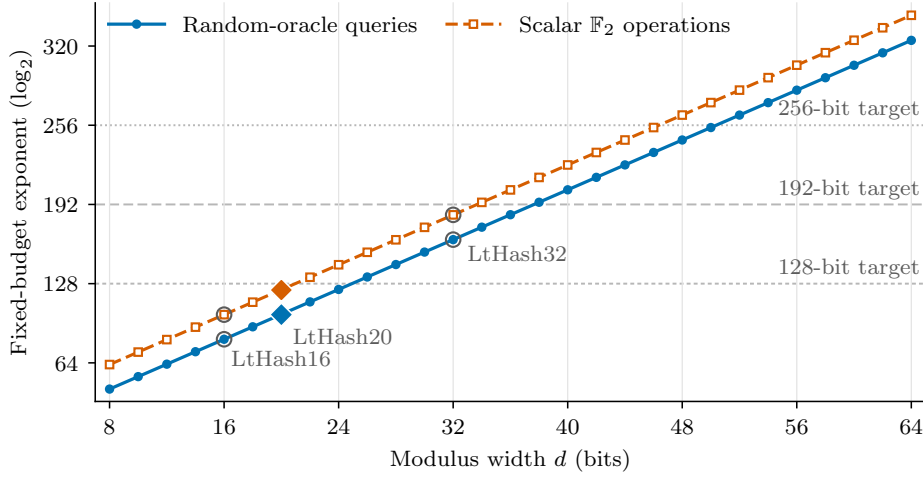


Fig. 1. Budgets ($p \geq 1/2$) across even d , with $N = 1024$ and $t = 550$ (LtHash20: $N = 1008$, $t = 542$).

input is no longer modeled as a set, but rather as a sequence of conventional message blocks.

In iSHAKE, an identifier block ID_i and a message payload M_i together form a message block. In fixed-size mode, blocks have equal length and identifiers encode their positions. In variable-size mode, block lengths may differ and each identifier contains a block nonce and a pointer to the next block [29, Section 4.2]. Let $G(M, ID) \in R_{64}^N$ denote the N consecutive 64-bit words obtained by parsing the $64N$ -bit output of $\text{SHAKE}(M \parallel ID)$. The hash of a message $((M_i, ID_i))_{i=1}^r$ is computed as $\sum_{i=1}^r G(M_i, ID_i)$. We model G as a random oracle.

Integrating Ternary SIS into TSG Attack. Our two-bit lifting attack can replace the single-list Wagner phase in the TSG attack [29, Section 4.2], yielding improved complexity for some parameters. Following the TSG attack, we choose two candidate blocks for each fixed ID and use the difference of their hash values as an input column for ternary collision search. The m candidate pairs (M_i^0, M_i^1) at identifiers ID_i should satisfy the following two conditions. (a). Every assignment $((M_i^{\sigma_i}, ID_i))_{i=1}^m$, for $\sigma \in \{0, 1\}^m$, must be a well-formed message. (b). The $2m$ strings $M_i^0 \parallel ID_i$ and $M_i^1 \parallel ID_i$, for $1 \leq i \leq m$, must be pairwise distinct. Set

$$y_i = G(M_i^0, ID_i) - G(M_i^1, ID_i).$$

The difference vectors y_i are independent and uniform in R_{64}^N . A nonzero ternary relation $z \in \{-1, 0, 1\}^m$ satisfying $\sum_i z_i y_i = 0$ can be transformed into two colliding messages. At identifier ID_i , the first and second messages use (M_i^0, M_i^1) if $z_i = 1$, (M_i^1, M_i^0) if $z_i = -1$, and the same candidate if $z_i = 0$. They are colliding messages because their hash difference is $\sum_i z_i y_i = 0$ and $z \neq 0$ makes them

distinct. For the iSHAKE case, forming an input vector requires two hashing oracle query, while for finite-set LtHash, it requires only one query.

Query Complexity Across Parameters. Our SIS-based TSG attack and the original Wagner-based TSG attack exhibit markedly different complexity profiles across parameter regimes. Let d be even. The leading query costs scale as $N^{d/2}$ for our two-bit lifting method and $2^{\sqrt{2Nd}}$ for original TSG [29, Section 3.2]. At fixed total output length Nd , TSG’s leading exponent is unchanged, whereas our bound $N^{d/2} = (Nd/d)^{d/2}$ grows with d .

Table 5 compares the two methods at the three power-of-two LtHash parameter sets and the four proposed iSHAKE configurations. The iSHAKE configurations use 64-bit coordinates and hence 32 lifting layers. For a limited-block configuration, $K_{\max}$ bounds the indexed blocks in either message. The iSHAKE128 and iSHAKE256 limits are 2^{25} and 2^{28} , respectively [22].

Table 5. Oracle-query costs ($\log_2$) at proposed parameters.

Configuration	(N, d)	Two-bit lifting ^E	TSG ^C
<i>Power-of-two LtHash (finite sets)</i>			
LtHash16	(1024, 16)	81.07	181.04
LtHash20	(1008, 20)	100.97	200.60
LtHash32	(1024, 32)	161.57	255.02
<i>Indexed-block iSHAKE</i>			
iSHAKE128, limited block*	(42, 64)	exceeds cap	109.52
iSHAKE128, unlimited block	(65, 64)	202.20	92.43
iSHAKE256, limited block*	(102, 64)	exceeds cap	235.14
iSHAKE256, unlimited block	(258, 64)	261.79	183.45

^E Expected queries. ^C Constant-success estimate. * Variable-size mode. “exceeds cap” refers to the evaluated query-optimal lifting schedule. Unlimited-block rows also allow identifiers of arbitrary length.

The two-bit lifting estimates are smaller at the three power-of-two LtHash parameter sets. For all four iSHAKE configurations, original TSG’s estimated query cost is below the expected cost of our two-bit lifting method, see concrete comparisons in Appendix A.4. Clearly, our SIS-based attack yields significant advantages under two distinct regimes. First, when the degree d of the modulus is fixed, our attack achieves polynomial-time complexity as $N \rightarrow \infty$, whereas the Wagner-based TSG method exhibits sub-exponential complexity. Second, when the product Nd is fixed, our attack remains efficient primarily for small modulus exponent d . A final distinction worth highlighting is that the original TSG attack holds over arbitrary moduli, whereas our variant is tailored specifically to power-of-two moduli endowed with rich algebraic structure.

6.5 Implications for LtHash Design

The power-of-two modulus that makes LtHash efficient also enables our recursive collision attacks, and for fixed d , increasing N raises their cost only polynomially.

We recommend large-prime moduli for future LtHash designs. For example, ZisK’s design for distributed challenge derivation already uses a lattice hash over the prime field of order $2^{64} - 2^{32} + 1$ [19]. The modulus and dimension should still be selected jointly against known lattice and generalized-birthday attacks.

Use of Generative AI OpenAI Codex and Kimi Code assisted with source organization and language editing. We reviewed and verified all such assistance.

References

1. Anza: Agave Accounts Lattice Hash implementation (2026), https://github.com/anza-xyz/agave/blob/e19ed2b9345c7d29f63a3da27749185cc1f28565/lattice-hash/src/lt_hash.rs, source code at commit e19ed2b9345c, accessed 22 August 2026
2. Bellare, M., Micciancio, D.: A new paradigm for collision-free hashing: Incrementality at reduced cost. In: Fumy, W. (ed.) EUROCRYPT’97. LNCS, vol. 1233, pp. 163–192. Springer, Berlin, Heidelberg (May 1997). https://doi.org/10.1007/3-540-69053-0_13
3. Bluesky Social: 0016 Permissioned Data (2026), <https://github.com/bluesky-social/proposals/blob/35a2d378297d9178561a5cc643c74a1262122266/0016-permissioned-data/README.md>, proposal, revision 3 September 2026, accessed 10 September 2026
4. BNB Chain: BNB Chain H2 2026 Tech Roadmap: Doubling down on speed (Jul 2026), <https://www.bnbchain.org/en/blog/bnb-chain-h2-2026-tech-roadmap-doubling-down-on-speed>, accessed 22 August 2026
5. Camion, P., Patarin, J.: The Knapsack hash function proposed at Crypto’89 can be broken. In: Davies, D.W. (ed.) EUROCRYPT’91. LNCS, vol. 547, pp. 39–53. Springer, Berlin, Heidelberg (Apr 1991). https://doi.org/10.1007/3-540-46416-6_3
6. Chailloux, A., Hermouet, P.: On the quantum equivalence between $S | \text{LWE} \rangle$ and ISIS . In: Heninger, N., Rosulek, M. (eds.) CRYPTO 2026. LNCS, vol. 16802, pp. 391–428. Springer, Cham (Aug 2026). https://doi.org/10.1007/978-3-032-35377-1_13, <https://arxiv.org/abs/2510.06097v2>, full version: arXiv:2510.06097v2
7. Chen, L., Chen, Y., Huang, Z., Sun, N., Yang, T., Zhang, Y.: Fiat-Shamir for bounded-depth adversaries. In: Dodis, Y. (ed.) ITC 2026. LIPIcs, vol. 385, pp. 4:1–4:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2026). <https://doi.org/10.4230/LIPIcs.ITC.2026.4>, <https://eprint.iacr.org/archive/2024/256/1708050016.pdf>, full version: IACR ePrint 2024/256 (16 February 2024)
8. Chen, Y., Liu, Q., Zhandry, M.: Quantum algorithms for variants of average-case lattice problems via filtering. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 372–401. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07082-2_14, https://doi.org/10.1007/978-3-031-07082-2_14

- [//eprint.iacr.org/2021/1093](https://eprint.iacr.org/2021/1093), full version: IACR ePrint 2021/1093 (6 October 2021)
9. Commonware: LtHash in `commonware-cryptography` v2026.7.1 (2026), <https://crates.io/crates/commonware-cryptography/2026.7.1>, release v2026.7.1, source commit `da15b3c08ac3`, accessed 22 August 2026
 10. Fraser, R.: `rsky-space`: atproto permissioned-data primitives (Aug 2026), <https://docs.rs/crate/rsky-space/0.4.1/source/>, version 0.4.1, accessed 15 September 2026
 11. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 197–206. ACM Press (May 2008). <https://doi.org/10.1145/1374376.1374407>, <https://eprint.iacr.org/2007/432>, full version: IACR ePrint 2007/432 (17 June 2010)
 12. Göös, M., Kamath, P., Sotiraki, K., Zampetakis, M.: On the complexity of modulo- q arguments and the Chevalley–Warning theorem. In: Saraf, S. (ed.) 35th Computational Complexity Conference (CCC 2020). LIPIcs, vol. 169, pp. 19:1–19:42. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2020). <https://doi.org/10.4230/LIPIcs.CCC.2020.19>
 13. Imran, M., Ivanyos, G.: Zero sum subsequences and hidden subgroups. *Quantum Information Processing* **23**(1), 14 (Jan 2024). <https://doi.org/10.1007/s11128-023-04228-2>
 14. Kothari, R., O’Donnell, R., Wu, K.: No exponential quantum speedup for SIS^∞ anymore. In: Bhaskara, A., Czumaj, A. (eds.) 58th ACM STOC. pp. 101–105. ACM, Salt Lake City, UT, USA (Jun 2026). <https://doi.org/10.1145/3798129.3800731>, <https://arxiv.org/abs/2510.07515v3>, full version: arXiv:2510.07515v3
 15. Lewi, K., Kim, W., Maykov, I.: Open-sourcing homomorphic hashing to secure update propagation (Mar 2019), <https://engineering.fb.com/2019/03/01/security/homomorphic-hashing/>, Meta Engineering, accessed 22 August 2026
 16. Lewi, K., Kim, W., Maykov, I., Weis, S.: Securing update propagation with homomorphic hashing. *Cryptology ePrint Archive*, Report 2019/227 (2019), <https://eprint.iacr.org/2019/227>
 17. Maitin-Shepard, J., Tibouchi, M., Aranha, D.F.: Elliptic curve multiset hash. *The Computer Journal* **60**(4), 476–490 (2017). <https://doi.org/10.1093/comjnl/bxw053>
 18. Marchan, L.E., Ordaz, O., Schmid, W.A.: Remarks on the plus-minus weighted Davenport constant. *International Journal of Number Theory* **10**(5), 1219–1239 (Aug 2014). <https://doi.org/10.1142/S1793042114500225>, <https://arxiv.org/abs/1308.3316>
 19. Massip, H.: Secure challenge derivation in ZisK (Oct 2025), <https://zisk.technology/secure-challenge-derivation-in-zisk/>, published 4 October 2025, accessed 15 September 2026
 20. Meta Platforms, Inc.: Folly: LtHash implementation (2026), <https://github.com/facebook/folly/blob/dea577df18b1cb35bafaf41764f2da57caabe5a1/folly/crypto/LtHash.h>, source code at commit `dea577df18b1`, accessed 22 August 2026
 21. Micciancio, D., Peikert, C.: Hardness of SIS and LWE with small parameters. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 21–39. Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_2, <https://www.iacr.org/archive/crypto2013/80420136/80420136.pdf>

22. Mihajloska, H., Gligoroski, D., Samardjiska, S.: Reviving the idea of incremental cryptography for the zettabyte era use case: Incremental hash functions based on SHA-3. In: Camenisch, J., Kesdoğan, D. (eds.) Open Problems in Network Security – iNetSec 2015. LNCS, vol. 9591, pp. 97–111. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39028-4_8, <https://eprint.iacr.org/2015/1028>, full version: IACR ePrint 2015/1028
23. Minder, L., Sinclair, A.: The extended k-tree algorithm. In: Mathieu, C. (ed.) 20th SODA. pp. 586–595. ACM-SIAM (Jan 2009). <https://doi.org/10.1137/1.9781611973068.65>
24. MISAKA-BTC: misakas: Source code and testnet documentation (2026), <https://github.com/MISAKA-BTC/misakas/tree/6fdf6ba71fb5599c6fd8a31868661531dbb79933>, source commit 6fdf6ba71fb5, accessed 15 September 2026
25. Nym Technologies SA: Nym directory and geolocation contracts (Sep 2026), <https://github.com/nymtech/nym/tree/7151978829fe52f3730fb09ad2e2177aa4e201c1/contracts>, release v2026.17-djibouti, source commit 7151978829fe, accessed 15 September 2026
26. Olson, J.E.: A combinatorial problem on finite Abelian groups, I. Journal of Number Theory **1**(1), 8–10 (Jan 1969). [https://doi.org/10.1016/0022-314X\(69\)90021-3](https://doi.org/10.1016/0022-314X(69)90021-3)
27. Prumo, B., Cesena, E., Siegel, J., Kim, S.: SIMD-0215: Homomorphic hashing of account state (2025), <https://github.com/solana-foundation/solana-improvement-documents/blob/20fe845aa165e7c51513ff0d39d7cb75f993fd9e/proposals/0215-accounts-lattice-hash.md>, SIMD-0215, activation commit 20fe845aa165, accessed 22 August 2026
28. Sei Protocol: Sei FlatKV LtHash, release v6.6.0 (2026), <https://github.com/sei-protocol/sei-chain/releases/tag/v6.6.0>, tagged source commit 082fbe7549e6, accessed 22 August 2026
29. Tang, L., Sun, Y., Gong, X.: On the regularity of the Generalized Birthday Problem. In: Heninger, N., Rosulek, M. (eds.) CRYPTO 2026. LNCS, vol. 16805, pp. 321–352. Springer, Cham (Aug 2026). https://doi.org/10.1007/978-3-032-35412-9_11, <https://eprint.iacr.org/archive/2025/1351/1780135842.pdf>, full version: IACR ePrint 2025/1351 (30 May 2026)
30. Trezy: HappyView v2.10 (Jun 2026), <https://happyview.dev/blog/happyview-2.10>, released 30 June 2026, accessed 15 September 2026
31. Vextaris: Vextaris Blockchain Technical Whitepaper (Jul 2026), <https://docs.vextaris.io/whitepaper/>, version 3.0, accessed 15 September 2026
32. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 288–304. Springer, Berlin, Heidelberg (Aug 2002). https://doi.org/10.1007/3-540-45708-9_19
33. WhatsApp: WhatsApp Encryption Overview (Feb 2026), https://www.whatsapp.com/security/WhatsApp_Security_Whitepaper_Preview.pdf, technical white paper, Version 9, updated 25 February 2026, accessed 23 August 2026

A Concrete Evaluation and Resource Accounting

We evaluate the query expressions, scalar-operation bounds, and peak storage reported in Section 6. Appendix A.4 gives the prior-work and iSHAKE comparisons. Displayed logarithms are rounded only after threshold comparisons.

A.1 Finite Rank Sums and Batch-Size Choices

For $0 \leq r \leq \min(a, b)$, a uniform binary a -by- b matrix has rank r with probability

$$P_{a,b}(r) = 2^{-ab} \prod_{i=0}^{r-1} \frac{(2^a - 2^i)(2^b - 2^i)}{2^r - 2^i}. \quad (29)$$

Conditioning on the rank in Equation (7) gives the finite sum

$$\gamma(a, b) = \sum_{r=0}^{\min(a,b)} P_{a,b}(r) 2^{r-a}. \quad (30)$$

Put $m = \min(a, b)$, $M = \max(a, b)$, and $H_n = \prod_{j=1}^n (1 - 2^{-j})$. The evaluator starts from

$$P_{a,b}(m) = \frac{H_M}{H_{M-m}}, \quad \frac{P_{a,b}(r-1)}{P_{a,b}(r)} = \frac{2^r - 1}{2^{r-1}(2^{a-r+1} - 1)(2^{b-r+1} - 1)} \quad (31)$$

and accumulates positive terms downward.

For $G \xleftarrow{\$} \mathbb{F}_2^{a \times b}$ and $1 \leq h \leq m$, the rank-tail bound

$$\Pr[\text{rk}(G) \leq m - h] < 4 \cdot 2^{-h(|a-b|+h)}$$

controls the omitted terms. Numerical estimates use 120-digit arithmetic. For the batch-size comparisons below, we use 60-digit interval arithmetic with outward rounding. We retain ranks $r \geq m - 24$ and bound the remaining probability mass by 2^{-623} . The supplementary code reproduces the estimates and certifies these comparisons.

The LtHash16 schedule $(550, \dots, 550)$ minimizes the expected query count because $t = 550$ minimizes each layer's affine cost map pointwise for $x \geq 1$. Fix $N = 1024$. The identity

$$\tilde{F}_{N,t}(x) = F_{N,t}(1) + (x - 1)\mu_{N,t} \quad (x \geq 1)$$

reduces pointwise comparison of the affine maps to comparison of $F_{N,t}(1)$ and $\mu_{N,t}$. We cover all positive integers t in three ranges.

For $1 \leq t \leq 511$, the bound $\gamma(a, b) \leq \min\{1, 2^{b-a}\}$ gives exact rational lower bounds that exclude every candidate.

For $512 \leq t \leq 611$, the interval calculation gives a unique minimum for both quantities at $t = 550$, with $t = 549$ second in both comparisons. The respective gaps exceed 3.4×10^{-4} and 0.067 . Every interval for either quantity has width below 8×10^{-54} .

For $t \geq 612$, the bounds $p_{N,t} \leq 1$ and $R_{N,t} \geq 512$ give

$$F_{N,t}(1) \geq r_N + 612, \quad \mu_{N,t} \geq 1124,$$

which exclude the infinite tail.

Thus $t = 550$ uniquely minimizes the affine map at every $x \geq 1$. Since the maps are strictly increasing, reverse induction through their composition shows that $(t_1, \dots, t_8) = (550, \dots, 550)$ uniquely minimizes expected queries among fixed positive-integer batch schedules. The scalar and storage analyses evaluate this selected schedule. For LtHash20 and LtHash32, the evaluator gives costs at the schedules displayed in Table 4.

A.2 Scalar-Operation Count

For even $d \geq 2$, consider $L_2 = d/2$ layers with a common batch size $t \geq 2$ and one terminal output. We factor each live basis with Gaussian elimination and solve each sign system directly, counting scalar operations as in Section 6.1. The bounds below count XORs and ANDs separately. We count lifting attempts, basis factorizations, accepted-output construction, and terminal sign propagation.

For one attempt with $K = k$, put $M = t + k$ and $r = \min\{N, M - 1\}$. The cost of an attempt and the additional cost of computing an accepted output column satisfy the following bounds on every run:

$$C_{\oplus}^{\text{att}}(k) = N(t - 1) + N(N - 1) + 3NM + \sum_{i=0}^{r-1} (N - i - 1)(M + 1 - i) + \binom{r}{2}, \quad (32)$$

$$C_{\otimes}^{\text{att}}(k) = NM, \\ C_{\oplus}^{\text{out}}(s, k) = 4NsM, \quad C_{\otimes}^{\text{out}}(s, k) = 2NsM \quad (s > 2). \quad (33)$$

The attempt bound includes parity aggregation, the basis solve, computation of the second-bit target, Gaussian elimination, and back substitution. At $s = 2$, the output value lies in R_0^N and needs no computation, so both output terms are zero.

After a success that consumes k basis columns, the expected cost of the next factorization is bounded by

$$C^{\text{reb}}(k) = \frac{N^2(N - 1)}{2} + N \sum_{j=1}^k \frac{N - j}{2^j - 1} \quad (k \geq 1), \\ C^{\text{reb}}(0) = 0. \quad (34)$$

The first term inserts the N independent live columns. At rank deficit j , the expected number of dependent candidates before the next rank increase is $(2^j - 1)^{-1}$, which gives the second term.

To average these costs, distinguish the law of K in an arbitrary attempt from its law after acceptance:

$$\omega_k = 2^{-N} \binom{N}{k}, \quad \pi_k = \frac{\omega_k \gamma(N - k, t - 1)}{p_{N,t}}.$$

Use ω for attempts and π for accepted outputs and refills. For $\diamond \in \{\oplus, \otimes\}$, set

$$\begin{aligned}\overline{C}_\diamond^{\text{att}} &= \sum_{k=0}^N \omega_k C_\diamond^{\text{att}}(k), \\ \overline{C}_\diamond^{\text{out}}(s) &= \sum_{k=0}^N \pi_k C_\diamond^{\text{out}}(s, k), \\ \overline{C}^{\text{reb}} &= \sum_{k=0}^N \pi_k C^{\text{reb}}(k).\end{aligned}$$

Count layers from the base. For $1 \leq j \leq L_2$, the expected output demands, input modulus exponents, and total expected attempt count are

$$\begin{aligned}m_j &= Q_{L_2-j}, \quad s_j = d - 2(j-1), \\ A_{L_2} &= \frac{1}{p_{N,t}} \sum_{j=1}^{L_2} m_j = \frac{1}{p_{N,t}} \sum_{r=0}^{L_2-1} Q_r.\end{aligned}$$

Terminal sign propagation depends on the number of selected base leaves. Put $\bar{k} = \sum_{k=0}^N \pi_k k$, and write $A_i = |\text{bsupp}(U_i)|$ for an input column U_i .

Lemma 6 (Expected Base-Support Size after One Layer). *Let the input columns U_i to $\text{TwoBitLayer}_{s,t}$ be fresh and base-support-disjoint, with $\mathbb{E}[A_i] = \sigma < \infty$ for every i . Every output W then satisfies*

$$\mathbb{E}[|\text{supp}(\text{rep}(W))|] = (t + \mathbb{E}[K \mid \text{success}])\sigma. \quad (35)$$

Proof. For one output, let $\mathcal{I}$ be the input indices consumed by its accepted attempt and T the number of inputs inspected. For fixed i, n , the event $E_{i,n} = \{i \in \mathcal{I}, T \leq n\}$ depends only on the first n input values modulo four. Freshness makes these values independent of their base supports, so $\mathbb{E}[A_i \mathbb{I}[E_{i,n}]] = \sigma \Pr[E_{i,n}]$. Since $T < \infty$ almost surely, this identity passes to the limit. Disjointness makes the output support size the sum of the consumed support sizes. The accepted attempt consumes $|\mathcal{I}| = t + K$ columns, and Lemma 3 gives the same distribution of K conditional on acceptance at every output position. Summing the nonnegative terms gives

$$\mathbb{E}[|\text{supp}(\text{rep}(W))|] = \sum_{i \geq 1} \mathbb{E}[A_i \mathbb{I}[i \in \mathcal{I}]] = \sigma \mathbb{E}[|\mathcal{I}|] = (t + \bar{k})\sigma.$$

□

Let σ_j be the expected base-support size of a level- j output. Starting from $\sigma_0 = 1$, the lemma gives

$$\sigma_j = (t + \bar{k})\sigma_{j-1} = (t + \bar{k})^j.$$

Thus the number Σ of base leaves in the terminal derivation satisfies $\mathbb{E}[\Sigma] = \sigma_{L_2} = (t + \bar{k})^{L_2}$. Its tree has fewer than $t\Sigma/(t-1)$ edges, which bounds the terminal sign-propagation cost.

Each layer factors its initial basis. After an output consumes basis columns, refilling and factorization are deferred until the next request. An output with $K = 0$ leaves the factorization unchanged. Apply Lemma 4 to the basis factorizations, lifting attempts, and output computations. Adding terminal sign propagation gives

$$\begin{aligned}\mathbb{E}[\mathcal{L}_\oplus] &\leq L_2 C^{\text{reb}}(N) + (\sum_j m_j - L_2) \bar{C}^{\text{reb}} \\ &\quad + A_{L_2} \bar{C}_\oplus^{\text{att}} + \sum_j m_j \bar{C}_\oplus^{\text{out}}(s_j) + \frac{t}{t-1} (t + \bar{k})^{L_2}, \\ \mathbb{E}[\mathcal{L}_\otimes] &\leq A_{L_2} \bar{C}_\otimes^{\text{att}} + \sum_j m_j \bar{C}_\otimes^{\text{out}}(s_j).\end{aligned}\tag{36}$$

The sum of the two right-hand sides in Equation (36) is the bound $\hat{L}_{N,d,t}$ used in the main text, so $\mathbb{E}[\mathcal{L}_{\mathbb{F}_2}] \leq \hat{L}_{N,d,t}$.

At $(N, d, t) = (1024, 16, 550)$, this sum gives Equation (27). The evaluations at $(1008, 20, 542)$ and $(1024, 32, 550)$ give the LtHash20 and LtHash32 entries in the concrete parameter table.

A.3 Bit-Sliced Layout and Peak Memory

Let $\mathcal{M}_{\text{peak}}$ be the peak storage in bits for the algebraic workspace and the signed derivation forest. We first count the workspace and forest edges, then choose records for the capped and uncapped executions.

Algebraic Workspace. For LtHash_{1024,16} with $t = 550$, the eight modulus exponents sum to $\sum_j s_j = 16 + 14 + \dots + 2 = 72$. In a compact bit-sliced layout, the column values, factorizations, and sign-system workspace use, respectively,

$$\begin{aligned}W_{\text{val}} &= (N + t) \left\lceil \frac{N}{64} \right\rceil \sum_j s_j, \\ W_{\text{fac}} &= 8N \left\lceil \frac{N}{64} \right\rceil, \\ W_{\text{sol}} &= N \left\lceil \frac{N + t + 1}{64} \right\rceil\end{aligned}$$

64-bit words. The triangular factor and its operation flags share one packed array, and accepted-output construction reuses a consumed trial slot. These three buffers total 1,969,920 words. A second solver buffer, four metadata words per stored-column slot, two pivot arrays per layer, and eight packed scratch vectors per layer add

$$W_{\text{ovh}} = W_{\text{sol}} + 4L_2(N + t) + 2L_2N + 8L_2 \left\lceil \frac{N}{64} \right\rceil = 93,376$$

words. The total of 2,063,296 words leaves more than two million bits of slack in

$$M_{\text{alg}} < 2^{27} \text{ bits.} \quad (37)$$

The algebraic workspace holds fewer than 13,000 root references, together with counters. This count includes their reserved 64-bit words. The two record layouts below account for the extra width.

Derivation Forest. Query indices reconstruct items from the base source's fixed sequence of distinct queries. We release oracle answers when their columns are consumed or discarded. We keep derivation records for accepted outputs in two append-only arrays per level. Let D_j count all level- j outputs ever returned, including those later discarded, and put $D_0 = \mathcal{Q}$. If E_{der} counts the forest's parent-to-child edges, disjoint consumption gives

$$tD_j \leq D_{j-1}, \quad E_{\text{der}} \leq \sum_{j \geq 1} D_{j-1} < \frac{t}{t-1} \mathcal{Q}. \quad (38)$$

Write $c_t = t/(t-1)$. Children of one node are contiguous, so an edge needs no parent pointer. One array stores child indices and signs. The other stores each internal node's first-edge offset and 11-bit arity, enough for $N + t = 1574$. Child indices identify queries at level one and outputs in the array below at higher levels.

Peak Storage under the Query Cap. Under $B_Q = \lceil 4Q_8 \rceil$, 84-bit indices and offsets suffice. Widening the stored references and counters adds less than 260,000 bits, within the slack in Equation (37). Fixed records use 85 bits per edge (index and sign) and 95 bits per internal node (offset and arity). There are at most E_{der}/t internal nodes, so $85 + 95/t < 96$ bits per edge cover both arrays. Thus

$$96 \frac{550}{549} B_Q + 2^{27} < 2^{89.66} \text{ bits.} \quad (39)$$

Expected Peak Storage. For the uncapped execution, put $w = \lceil \log_2(c_t \mathcal{Q}) \rceil$. A child record uses a w -bit index and one sign. The arrays are paged, so ordinary appends do not copy prior records. When w increases, storing the old and new records takes $2w + 1$ bits per edge and $2w + 21$ bits per internal node. Counting both copies gives, for every run,

$$\begin{aligned} M_{\text{der}} &< E_{\text{der}} \left(2w + 1 + \frac{2w + 21}{t} \right) \\ &\leq c_t \mathcal{Q} \left[2 \left(1 + \frac{1}{t} \right) \log_2(c_t \mathcal{Q}) + 3 + \frac{23}{t} \right]. \end{aligned} \quad (40)$$

The term $\mathcal{Q} \log_2 \mathcal{Q}$ requires a second-moment estimate for the query count. Define

$$\begin{aligned}\nu_k &= \sum_{j=1}^k \frac{2^{-j}}{(1-2^{-j})^2}, & V_I &= \nu_N, \\ V_T &= \frac{t^2(1-p_{N,t})}{p_{N,t}^2}, \\ V_R &= \sum_{k=0}^N \pi_k(\nu_k + r_k^2) - R_{N,t}^2, \\ \alpha &= V_T + V_R, & \kappa &= V_I - V_R.\end{aligned}$$

For m outputs, let $\mathcal{C}(m)$ count the input columns requested by the layer. The initial basis fill has variance V_I . The trial columns for m outputs contribute variance V_T per output, and each of the $m-1$ refills contributes variance V_R . Freshness and the basis-independent attempt law make these contributions independent: the geometric retry count is independent of the accepted K . Hence

$$\text{Var}(\mathcal{C}(m)) = V_I + mV_T + (m-1)V_R = \alpha m + \kappa.$$

If V_ℓ is the variance of the number of base columns requested after ℓ composed layers, independence for each fixed output count and the law of total variance give

$$\begin{aligned}V_0 &= 0, \\ V_{\ell+1} &= \mu_{N,t}^2 V_\ell + \alpha Q_\ell + \kappa, & V_8 &= \text{Var}(\mathcal{Q}).\end{aligned}\tag{41}$$

Put $M_8 = V_8 + Q_8^2 = \mathbb{E}[\mathcal{Q}^2]$. Jensen's inequality, with probability weights proportional to $\mathcal{Q}$, gives

$$\mathbb{E}[\mathcal{Q} \log_2 \mathcal{Q}] \leq Q_8 \log_2 (M_8 / Q_8).$$

Jensen's inequality also gives $\mathbb{E}[w] < 83$, so the expected extra storage for these references and counters fits the same 260,000-bit allowance used above for $w \leq 84$. Combining these bounds with Equations (37) and (40) yields

$$\mathbb{E}[\mathcal{M}_{\text{peak}}] < 2^{88.45} \text{ bits}.\tag{42}$$

An in-place traversal propagates signs and streams the two output sets, freeing visited records.

A.4 Derivation of Comparison Costs

Finite-Set LtHash. Under the post-order evaluation with index trimming stated in Section 6.2, the dominant terms in the peak-space estimates are

$$\begin{aligned}\text{CPW: } & 2^{161} 3^{64} \approx 2^{262.44}, \\ \text{MS: } & 2^{127} \cdot 2^{(16384 - 128 \log_2 3)/126} \approx 2^{255.42}.\end{aligned}$$

These peak-space evaluations are in bits and use the expected list sizes in the cited analyses.

For TSG, put

$$b = Nd, \quad k = \lfloor \sqrt{b/2 + 1} \rfloor, \quad M = 2^{b/(k+1)+1}.$$

Applying the single-list algorithm directly to finite-set hash outputs uses M queries per run, giving query exponents 181.04, 200.60, and 255.02 at the three power-of-two LtHash parameter sets, respectively. For LtHash_{1024,16}, its $O(kM)$ time and optimized $O(bM)$ -bit memory bounds [29, Theorems 4–5 and Appendix D.2] give

$$\begin{aligned} T_{\text{TSG}} &\approx 2^{187.54}, \\ M_{\text{TSG}} &\approx 2^{195.04} \text{ bits.} \end{aligned}$$

For the Regev–CLZ one-bit recursion, we evaluate the implementation that uses 0/1 representatives in each local relation modulo two. Put $r = N + 1 = 1025$ and $J = (r^{16} - 1)/(r - 1)$. One block uses fewer than r^3 scalar $\mathbb{F}_2$ operations. A depth-first traversal retaining the bits that encode selected columns for collision recovery uses rJ bits. Including the final traversal and solver workspace gives

$$\begin{aligned} T_{\text{1bit}} &< (r^3 + r)J < 2^{180.03}, \\ M_{\text{1bit}} &< rJ + 2^{29} < 2^{160.03} \text{ bits.} \end{aligned}$$

iSHAKE Schedules and Query Counts. Since the affine map $\tilde{F}_{N,t}$ in Equation (12) is increasing, we can optimize fixed positive-integer batch schedules from the terminal layer toward the base. Before imposing the block caps, the optimal 32-layer schedules give

Configuration	Schedule $(t_{32}, \dots, t_1)$	$\log_2 \mathbb{E}[\text{pairs}]$
iSHAKE128 limited	(27, 28, ..., 28)	183.24
iSHAKE128 unlimited	(40, 41, ..., 41)	201.20
iSHAKE256 limited	(61, 62, ..., 62)	220.30
iSHAKE256 unlimited	(146, 147, ..., 147)	260.79

Each pair requires two SHAKE evaluations. Adding one to the expected pair exponents gives the lifting values for the unlimited-block configurations.

For the two limited-block configurations, the optimized schedules have deterministic support lower bounds $2^{153.78}$ and $2^{190.51}$. These lower bounds exceed the respective block limits $K_{\max} = 2^{25}$ and 2^{28} .

For TSG, these configurations give

Configuration	k	Initial list size ($\log_2$)
iSHAKE128 limited	24	108.520
iSHAKE128 unlimited	45	91.435
iSHAKE256 limited	27	234.143
iSHAKE256 unlimited	90	182.451

Adding one to these exponents accounts for the two SHAKE evaluations and gives the query entries in Table 5. The two limited-block entries use the variable-size mode described in Section 6.4 [29, Table 2].

TSG Query Estimates versus Fixed Lifting Schedules. At all four iSHAKE configurations, TSG’s query estimate is below the expected cost of every fixed lifting schedule satisfying the block constraints. Monotonicity of acceptance in K gives $R_{N,t} \geq \mathbb{E}[K \mid \text{success}] \geq N/2$. With $p_{N,t} \leq 1$ and $c_{N,t} = r_N - R_{N,t} \geq 0$, each layer satisfies

$$\tilde{F}_{N,t}(x) \geq (N/2 + t)x.$$

Every 32-layer schedule therefore requires at least $(N/2 + 1)^{32}$ candidate pairs in expectation, placing both unlimited-block iSHAKE configurations above their TSG entries. For limited-block configurations, each layer multiplies the deterministic support lower bound by at least t_j , so a necessary condition is

$$\prod_{j=1}^{32} t_j \leq K_{\max}.$$

If r layers have $t_j \geq 3$, then $3^r \leq K_{\max}$. Consequently, at least 17 layers for iSHAKE128 and 15 for iSHAKE256 have $t_j \in \{1, 2\}$. For these batch sizes,

$$p_{N,1} = (3/4)^N, \quad p_{N,2} = 2(3/4)^N - (5/8)^N < 2(3/4)^N,$$

so $t_j/p_{N,t_j} \geq (4/3)^N$ for each such layer. Rejecting outputs whose supports exceed the block cap can only increase the expected number of candidate pairs. Including the two SHAKE evaluations per pair, the resulting lower bounds on expected queries for fixed schedules that satisfy the block limits are approximately $2^{297.34}$ and $2^{636.01}$, respectively.