

Tree Encodings III: Time/Space Hardness from Lattice Problems

Damiano Abram*

Giulio Malavolta[†]

Lawrence Roy[‡]

Abstract

Assuming the polynomial-time hardness of a variant of the short integer solution (SIS) problem, we show the existence of sequential and memory-hard functions. Such an assumption postulates the hardness of a problem against polynomial-time algorithms, regardless of their memory and depth. Yet, we show that this assumption implies that $P \neq NC$, that $P \neq L$, and that the NC hierarchy is proper, i.e., that $NC^1 \subsetneq NC^2 \subsetneq NC^3 \dots$

Our main technical tool is a new variant of homomorphic computation over lattice encodings, which does not impose a bound on the circuit depth and is not based on bootstrapping.

1 Introduction

A family of functions $\{f_T\}$ is called *sequential* if any algorithm to evaluate f_T requires $\Omega(T)$ sequential steps. Conversely, we say that a function family $\{f_S\}$ is *memory-hard* if evaluating f_S requires $\Omega(S)$ space.

Sequential and memory-hard functions play a central role in modern cryptography: They underlie the design of delay mechanisms such as time-lock puzzles [RSW96] verifiable delay functions [BBBF18], proofs of work [DN93], proofs of space [DFKP15], password hashing schemes [ACP⁺17, JM21], and key-derivation functions [Per09], to mention a few. More broadly, the cryptographic hardness of a particular resource (be it space or time) is a natural way to leverage the power of cryptographic assumptions when the budget of the adversary is constrained in some way.

In the broader complexity-theoretic landscape, the existence of sequential functions can be thought of as the average case version of the P vs NC question, i.e., the existence of non-parallelizable languages. It is also indirectly connected to the class of total search problems $PPAD$ [Pap94b]: Many natural problems in $PPAD$ can be thought of as the task of following an implicitly described directed path, and hardness results [BPR15] often boil down to showing that one cannot find “shortcuts” for such problems. For instance, sequential problems have been used [CHK⁺19, LV20, BCH⁺22] to establish the cryptographic hardness of finding a Nash equilibrium. On the other hand, memory-hard functions are the average-case version of the P vs L problem, and they are intimately connected with *pebbling complexity* [LT82, AS15] on graphs, where one seeks to understand the amount of memory required to carry out a computation efficiently. Indeed, many of the candidate constructions of memory-hard functions are based on graph-pebbling phenomena [AS15, CP18, DLM19], and their security analyses often amount to proving lower bounds on the cumulative memory or space-time cost of evaluating the underlying graph [ABP17].

*University of Edinburgh

[†]Bocconi University

[‡]IBM Research

Despite this intuitive appeal, proving these kinds of properties for explicit functions remains difficult. At present, there are only a handful of candidate sequential, such as repeated squaring [RSW96] or repeated hashing [MMV11], or memory-hard [Per09, JM21] functions, and the evidence for their security is mostly empirical. While proving unconditional results might be out of the reach of current techniques, one could still hope for more substantial evidence. Contrast this with “regular” cryptographic assumptions, such as the learning with errors (LWE) problem [Reg05], whose hardness is supported by worst-case-to-average case reductions [Reg05, Pei09] and NP-hardness in certain parameter regimes [Ajt98] (although not those useful for cryptographic applications).

Moreover, the world of sequentiality/memory-hardness assumptions is almost entirely divorced from the landscape of traditional “one-way” problems in cryptography, and hard problems in either world tend to be quite different. This is perhaps not surprising: There is no a priori reason why a one-way function (i.e., hardness against all polynomial time adversaries) should bear any implication in the setting where admissible attackers run in polynomial time, but they are bounded in depth or space. In complexity theory, this is analogous to the status of the P vs NP problem compared to the P vs NC problem and the P vs L problem. For example, with current knowledge, it is (perhaps unlikely but) in principle possible that $P \neq NP$ but $P = NC$ or that $P = NP$ but $P \neq NC$.

1.1 Our Results

We show how to connect “regular” lattice problems in cryptography, with the existence of sequential and memory-hard functions. Our starting point is the Decomposed Short-Integer Solution (SIS) problem: Let

$$\mathbf{Z} := [\mathbf{A}_0 \cdot \mathbf{B} - \mathbf{u}_0^\top \otimes \mathbf{G} \quad \cdots \quad \mathbf{A}_{t-1} \cdot \mathbf{B} - \mathbf{u}_{t-1}^\top \otimes \mathbf{G}]$$

where $\{\mathbf{A}_i\}$ are uniformly sampled, $\{\mathbf{B}_j\}$ are sampled from a discrete Gaussian, $\mathbf{G}$ is the gadget matrix [MP12] and $\{\mathbf{u}_i\}$ is the standard basis of $\mathbb{R}^t$. The Decomposed SIS problem postulates that no polynomial-time algorithm can compute a *short* vector $\mathbf{w}$ such that

$$\mathbf{Z} \cdot \mathbf{w} = \mathbf{0}.$$

The decomposed SIS problem can be thought of as a *structured* version of the regular SIS problem [Reg05] (SIS is defined analogously but for a random matrix $\mathbf{Z}$) and it is the natural *search* analogue of the decomposed LWE problem [AMR25a], which has been the subject of recent intense activities in the cryptographic community [Wee25, AMR25a, WW25, CHW25, LLL25]. It can be shown that the Decomposed SIS problem is at least as hard as the Decomposed LWE problem, which in turn is at least as hard as the Succinct LWE problem [Wee25]. Notice that Decomposed SIS is a “traditional” one-wayness assumption, in the sense that it postulates hardness against any polynomial-time algorithm, regardless of their memory or circuit depth.

First, we show that, if the Decomposed SIS problem is hard, then for any S there exists a family of functions that cannot be computed in polynomial time with less than S space. In particular, this implies the following.

Theorem 1.1 (Memory-Hardness). *If the Decomposed SIS problem is hard, then $P \neq L$.*

We show an analogous statement for depth-hardness, also assuming Decomposed SIS: For any T , there exists a family of functions that cannot be computed in polynomial time with depth less than T . A consequence of this fact is the following theorem.

Theorem 1.2 (Sequentiality). *If the Decomposed SIS problem is hard, then $P \neq NC$.*

We stress that the new point of [Theorem 1.1](#) and [Theorem 1.2](#) is not that a cryptographic assumption implies a separation between two complexity classes (for instance it is tautological that the hardness of Decomposed SIS implies that $P \neq NP$), but rather that one can prove memory-hardness and sequentiality from an assumption that does not distinguish between small or large memory/depth – beyond the fact that it is polynomially bounded. At the cost of stating the obvious, nothing to the effect of $P \neq NP \implies P \neq L$ or $P \neq NP \implies P \neq NC$ is currently known. We should also note that Decomposed SIS is an *average-case* problem, so one could expect even stronger average-case results for memory-hardness and sequentiality. Indeed our results imply these, and we refer the reader to [Corollary 4.4](#) and [Corollary 4.6](#) for precise statements.

Finally, we show that a careful analysis of our sequential function allows us to establish that the NC hierarchy is proper.

Theorem 1.3 (NC hierarchy). *If the Decomposed SIS problem is hard, then $NC^j \subsetneq NC^{j+1}$ for all $j \geq 1$.*

Our main technical innovation is a new construction of homomorphic lattice encodings [\[BGG⁺14\]](#) for space-bounded Turing machines. Contrary to prior work [\[BGG⁺14, AMR25b, AMR25a\]](#) our lattice encodings support *depth unbounded* computations, and do so without bootstrapping. The only other lattice encoding scheme that does not fix a bound on the depth is based on bootstrapping [\[HLL23\]](#). Besides requiring a circularity-type assumption, bootstrapping introduces a non-linear step in the evaluation procedure, which makes the scheme unsuitable for our reductions.

2 Preliminaries

2.1 Notation and Linear Algebra

The security parameter is λ . All unspecified polynomial bounds are in λ . We write $[a] = \{0, \dots, a-1\}$ and use column vectors. Matrices and vectors over $\mathbb{Z}_q$ are identified with their centered integer representatives whenever a norm is taken. The notation $x \stackrel{\$}{\leftarrow} X$ means that x is sampled from the distribution X .

For a matrix $\mathbf{M}$, define

$$\|\mathbf{M}\|_{\infty}^{\infty} := \max_{\|\mathbf{v}\|_{\infty}=1} \|\mathbf{M}\mathbf{v}\|_{\infty} = \max_i \sum_j |M_{i,j}|.$$

For this norm, we have

- $\|\mathbf{AB}\|_{\infty}^{\infty} \leq \|\mathbf{A}\|_{\infty}^{\infty} \|\mathbf{B}\|_{\infty}^{\infty}$,
- $\|\mathbf{A} \otimes \mathbf{B}\|_{\infty}^{\infty} = \|\mathbf{A}\|_{\infty}^{\infty} \|\mathbf{B}\|_{\infty}^{\infty}$, and
- $\|[\mathbf{A} \mid \mathbf{B}]\|_{\infty}^{\infty} \leq \|\mathbf{A}\|_{\infty}^{\infty} + \|\mathbf{B}\|_{\infty}^{\infty}$.

We use column-wise vectorization, so

$$\text{vec}(\mathbf{ABC}) = (\mathbf{C}^{\top} \otimes \mathbf{A})\text{vec}(\mathbf{B}). \tag{1}$$

Let $q = 2^L$ and set $n = kL$. With

$$\mathbf{g}^{\top} = (1, 2, \dots, 2^{L-1}) \quad \text{and} \quad \mathbf{G} = \mathbf{I}_k \otimes \mathbf{g}^{\top} \in \mathbb{Z}_q^{k \times n},$$

the map $\mathbf{G}^{-1}$ denotes canonical binary decomposition using representatives in $\{0, \dots, q-1\}$, extended columnwise to matrices. Norms, by contrast, use centered representatives. Hence

$$\mathbf{G}\mathbf{G}^{-1}(\mathbf{Y}) = \mathbf{Y} \quad (2)$$

for every matrix $\mathbf{Y}$ with k rows. If $\mathbf{Y}$ is uniform over $\mathbb{Z}_q^{k \times r}$, then $\mathbf{G}^{-1}(\mathbf{Y})$ is uniform over $\{0, 1\}^{n \times r}$. Moreover,

$$\|\mathbf{G}^{-1}(c\mathbf{G})\|_\infty^\infty \leq n \quad (c \in \mathbb{Z}_q). \quad (3)$$

2.2 Lattices, Discrete Gaussians, and Decomposed SIS

For $\mathbf{C} \in \mathbb{Z}_q^{a \times b}$, its q -ary kernel lattice is

$$\Lambda_q^\perp(\mathbf{C}) := \{\mathbf{z} \in \mathbb{Z}^b : \mathbf{C}\mathbf{z} = \mathbf{0} \pmod{q}\}.$$

For $\sigma > 0$, the discrete Gaussian $D_{\mathbb{Z}, \sigma}$ assigns mass proportional to $\exp(-\pi x^2/\sigma^2)$ to $x \in \mathbb{Z}$. We write $D_\sigma^{a \times b}$ for the product distribution on integer matrices. For $\sigma = \omega(\sqrt{\log \lambda})$, a union bound and the standard discrete-Gaussian tail estimate imply that a polynomial number of samples from $D_{\mathbb{Z}, \sigma}$ all have magnitude at most $\sigma\sqrt{\lambda}$ except with probability $\delta_{\text{tail}}(\lambda) := 2^{-\Omega(\lambda)}$ [Ban93]. All norm claims below are conditioned on this overwhelming-probability event.

Fix $t = 2mn$. The structured matrix distribution used throughout the paper is the following.

Definition 2.1 (Decomposed sampler). $\text{DSample}(1^k, 1^m, 1^t, q, \sigma)$ samples

$$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{tk \times m}, \quad \mathbf{B} \xleftarrow{\$} D_\sigma^{m \times tn}.$$

Write $\mathbf{A}_i \in \mathbb{Z}_q^{k \times m}$ for the i -th block of k rows of $\mathbf{A}$, and let $\mathbf{u}_i$ be the i -th standard basis vector of $\mathbb{Z}^t$. It returns $(\mathbf{A}, \mathbf{B}, \mathbf{Z})$, where

$$\mathbf{Z} := [\mathbf{Z}_0 \mid \dots \mid \mathbf{Z}_{t-1}] \in \mathbb{Z}_q^{k \times nt^2} \quad \text{and} \quad \mathbf{Z}_i := \mathbf{A}_i \mathbf{B} - \mathbf{u}_i^\top \otimes \mathbf{G} \in \mathbb{Z}_q^{k \times tn}.$$

Every public occurrence of $\mathbf{B}$ uses its residue class modulo q , with each entry in the canonical fixed-length encoding. The sampled integer matrix is retained only as a lift for the Gaussian tail and norm estimates. We recall the following variant of the decomposed SIS problem [AMR26b].

Definition 2.2 (Extended decomposed SIS). Let $r = r(\lambda)$ and $\beta = \beta(\lambda)$. In the $(k, m, t, q, \sigma, r, \beta)$ -ExtDSIS problem, the challenger samples

$$(\mathbf{A}, \mathbf{B}, \mathbf{Z}) \xleftarrow{\$} \text{DSample}(1^k, 1^m, 1^t, q, \sigma), \quad \mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{k \times r},$$

and gives $(\mathbf{A}, \mathbf{B}, \mathbf{Z}, \mathbf{U})$ to the adversary. The adversary wins if it returns

$$\mathbf{0} \neq \mathbf{w} \in \mathbb{Z}^{nt^2+r}, \quad [\mathbf{Z} \mid \mathbf{U}]\mathbf{w} = \mathbf{0} \pmod{q}, \quad \|\mathbf{w}\|_\infty \leq \beta.$$

In [AMR26b] it is shown that extended decomposed SIS is equivalent, with polynomial parameter loss, to decomposed SIS, and that decomposed SIS is at least as hard as decomposed LWE [AMR26b]. We therefore state reductions to Definition 2.2 and invoke ordinary decomposed SIS in the qualitative corollaries.

2.3 Computational Models

We use a fixed universal deterministic random-access machine U with read-only input memory, read-only program memory, and read-write work memory. A program memory is a bit string p of length at most 2^d , identified with the lookup function $p : \{0, 1\}^d \rightarrow \{0, 1\}$ appropriately padded with 0s. A d -bit configuration $c \in \{0, 1\}^d$ contains the control state and program counter, the address registers, and the work memory. The control state has a designated halting state, and the configuration has a designated output bit, denoted by $\text{out}(c)$. We use the standard convention that a halting configuration is a fixed point of the transition function.

For a fixed read-only input x , let C_x denote the one-step transition circuit of U with the answers to its program-memory reads left as input wires. For a program memory p , let

$$C_x^p : \{0, 1\}^d \rightarrow \{0, 1\}^d$$

be the transition map obtained by answering a program-memory read at address β with $p(\beta)$. We assume that one transition makes a constant number of random-access reads and that, after fixing the answers to these reads as input wires, C_x has bounded-fan-in depth $O(\log d)$ and polynomial size. If the input, output index, and program addresses do not fit in d bits, we replace d by

$$d' = O(d + \log |x| + \log n + \log |\mathcal{M}|).$$

Our statements use the compiled configuration length d' and rename it d .

The class NC^j consists of languages decided by logspace-uniform, polynomial-size, bounded-fan-in Boolean circuits of depth $O(\log^j N)$ and $\text{NC} = \bigcup_{j \geq 1} \text{NC}^j$. In a slight abuse of notation, we say that a function f is computable by an NC^j circuit if the language

$$\{(x, i) : i \in [|f(x)|], f(x)_i = 1\},$$

with i encoded in binary, belongs to NC^j .

2.4 Matrix Branching Programs

A length- ℓ read-once matrix branching program is a collection of matrices

$$\text{BP} = \{\mathbf{M}_{i,b} \in \mathbb{Z}_q^{w_i \times w_{i+1}} : i \in [\ell], b \in \{0, 1\}\}$$

whose value on $\alpha \in \{0, 1\}^\ell$ is

$$\text{BP}(\alpha) = \prod_{i=0}^{\ell-1} \mathbf{M}_{i, \alpha_i}.$$

The *width* of a branching program is $\max_i w_i$, and its *transition norm* is $\max_{i,b} \|\mathbf{M}_{i,b}\|_\infty$. We say that a branching program is *scalar-output* if $w_0 = w_\ell = 1$ and *matrix-output* if $w_0 = w_\ell = n$.

2.5 Correlation Intractability

We recall the notions of correlation intractability and searchable relations.

Definition 2.3 (Searchable relation). *A relation ensemble $\mathcal{R} = (\mathcal{R}_\lambda)_\lambda$ with $\mathcal{R}_\lambda \subseteq \mathcal{X}_\lambda \times \mathcal{Y}_\lambda$ is searchable if for every x there is at most one y such that $(x, y) \in \mathcal{R}_\lambda$. We write $\mathcal{R}_F$ for the graph relation*

$$\mathcal{R}_F = \{(x, y) : y = F_\lambda(x)\}.$$

Definition 2.4 (Correlation intractability). *A keyed hash family $(\text{KeyGen}, \text{CHash})$ is correlation intractable for $\mathcal{R}$ with time bound s and error ε if every adversary of running time at most $s(\lambda)$ satisfies*

$$\Pr \left[(x, \text{CHash}(\text{hk}, x)) \in \mathcal{R}_\lambda : \text{hk} \xleftarrow{\$} \text{KeyGen}(1^\lambda), x \leftarrow \mathcal{A}(\text{hk}) \right] \leq \varepsilon(\lambda).$$

When omitted, s is polynomial and ε is negligible.

3 Lattice Encodings

We first recall the matrix identities and homomorphic evaluation algebra from [AMR26a]. We then give the new homomorphic read operation and derive the new key equation.

3.1 Background on Lattice Encodings

We recall some background material on the encodings from [AMR26a]. Let $t = 2mn$, and let

$$\text{pk} = (\mathbf{A}, \mathbf{B}, \mathbf{Z}) \xleftarrow{\$} \text{Setup}(1^\lambda, 1^t) := \text{DSample}(1^{k(\lambda)}, 1^{m(\lambda)}, 1^t, q(\lambda), \sigma(\lambda))$$

as in Definition 2.1. Define

$$\text{Hash}(\text{pk}, \mathbf{x}) := \text{vec}(\mathbf{B}(\mathbf{x} \otimes \mathbf{I}_n)) \in \mathbb{Z}_q^{mn}, \quad \mathbf{x} \in \mathbb{Z}_q^t. \quad (4)$$

This map is linear over $\mathbb{Z}_q$. We write

$$\text{Hash}(\text{pk}, \mathbf{x}) = \mathbf{D}\mathbf{x}, \quad \mathbf{D} = [\mathbf{D}_0 \mid \mathbf{D}_1] \in \mathbb{Z}_q^{mn \times 2mn},$$

where $\mathbf{D}_0, \mathbf{D}_1 \in \mathbb{Z}_q^{mn \times mn}$. The following establishes the expansion of a hash encoding into an encoding of its preimage [AMR26a].

Lemma 3.1 (Encoding expansion). *There are deterministic polynomial-time algorithms*

- $\text{ExpandM}(\text{pk}) \mapsto \mathbf{P} \in \mathbb{Z}_q^{mn^2 \times tn}$,
- $\text{ExpandC}(\text{pk}, \mathbf{x}) \mapsto \mathbf{Q}_\mathbf{x} \in \mathbb{Z}_q^{nt^2 \times tn}$

such that

$$[\mathbf{G}(\text{Hash}(\text{pk}, \mathbf{x})^\top \otimes \mathbf{I}_n) \mid \mathbf{Z}] \begin{bmatrix} \mathbf{P} \\ \mathbf{Q}_\mathbf{x} \end{bmatrix} = \mathbf{G}(\mathbf{x}^\top \otimes \mathbf{I}_n). \quad (5)$$

Moreover, $\|\mathbf{P}\|_\infty^\infty \leq t$ and $\|\mathbf{Q}_\mathbf{x}\|_\infty^\infty \leq \|\mathbf{x}\|_\infty$.

Writing $\mathbf{P} = [\mathbf{P}_0 \mid \mathbf{P}_1]$ and $\mathbf{Q}_\mathbf{x} = [\mathbf{Q}_{\mathbf{x},0} \mid \mathbf{Q}_{\mathbf{x},1}]$, we have $\mathbf{P}_b \in \mathbb{Z}_q^{mn^2 \times mn^2}$ and $\mathbf{Q}_{\mathbf{x},b} \in \mathbb{Z}_q^{nt^2 \times mn^2}$. The Merkle extension of Equation (4) hashes pairs of mn -dimensional child vectors. For a scalar-output BP, let

$$\mathbf{h}_{\text{BP}} := \text{MerkleHash}(\text{pk}, \{\text{BP}(\beta)\mathbf{e}_0 : \beta \in \{0, 1\}^\ell\}), \quad \mathbf{e}_0 = (1, 0, \dots, 0)^\top \in \mathbb{Z}_q^{mn}.$$

Although the tree has 2^ℓ leaves, it is shown in [AMR26a] that its root is computable without expanding the tree: If $\text{BP}(\beta) = \prod_i \mathbf{M}_{i,\beta_i}$, then

$$\mathbf{h}_{\text{BP}} = \left(\prod_{i=0}^{\ell-1} (\mathbf{D}_0 \otimes \mathbf{M}_{i,0} + \mathbf{D}_1 \otimes \mathbf{M}_{i,1}) \right) \mathbf{e}_0. \quad (6)$$

The following lemma from [AMR26a] shows that the root of the hash can be efficiently expanded to any of its leaves.

Lemma 3.2 (Branching-program expansion). *For every scalar-output BP and $\alpha \in \{0, 1\}^\ell$, one can compute $\mathbf{Q}_{\text{BP}, \alpha} \in \mathbb{Z}_q^{nt^2 \times n}$ in time polynomial in λ, ℓ , and the width of BP, such that*

$$[\mathbf{G}(\mathbf{h}_{\text{BP}}^\top \otimes \mathbf{I}_n) \mid \mathbf{Z}] \begin{bmatrix} \left(\prod_{i=0}^{\ell-1} \mathbf{P}_{\alpha_i} \right) (\mathbf{e}_0 \otimes \mathbf{I}_n) \\ \mathbf{Q}_{\text{BP}, \alpha} \end{bmatrix} = \text{BP}(\alpha) \mathbf{G}. \quad (7)$$

Furthermore,

$$\|\mathbf{Q}_{\text{BP}, \alpha}\|_\infty^\infty \leq \max_\beta |\text{BP}(\beta)| \sum_{a=0}^{\ell-1} (t \|\mathbf{D}\|_\infty^\infty)^a. \quad (8)$$

The following procedure *normalizes* the branching programs into a canonical form. Let $\mathbf{R}_{\text{Can}} \in \{0, 1\}^{mn^4 \times n}$ be the fixed matrix whose block indexed by $(r, s) \in [n]^2$ is $(\mathbf{e}_0 \otimes \mathbf{I}_n) \mathbf{e}_r \mathbf{e}_s^\top \in \{0, 1\}^{mn^2 \times n}$, where $\mathbf{e}_0 \in \mathbb{Z}_q^{mn}$ and $\mathbf{e}_r, \mathbf{e}_s \in \mathbb{Z}_q^n$. For $\alpha \in \{0, 1\}^\ell$, define

$$\mathbf{P}_\alpha := \left(\prod_{i=0}^{\ell-1} (\mathbf{I}_{n^2} \otimes \mathbf{P}_{\alpha_i}) \right) \mathbf{R}_{\text{Can}} \in \mathbb{Z}_q^{mn^4 \times n}. \quad (9)$$

Then

$$\|\mathbf{P}_\alpha\|_\infty^\infty \leq t^\ell. \quad (10)$$

For a bit matrix $\mathbf{M} \in \{0, 1\}^{n \times mn^4}$, write $\text{Can}(\mathbf{M})(\alpha) := \mathbf{M} \mathbf{P}_\alpha$. The following is proven in [AMR26a].

Lemma 3.3 (Normalization). *Let BP be a length- ℓ matrix-output read-once branching program. There are deterministic algorithms, polynomial in the public parameters, ℓ , and the width of BP, that output $\mathbf{M}_{\text{BP}} \in \{0, 1\}^{n \times mn^4}$ and $\mathbf{H}_{\text{BP}, \alpha} \in \mathbb{Z}_q^{nt^2 \times n}$ such that*

$$\mathbf{G} \text{Can}(\mathbf{M}_{\text{BP}})(\alpha) + \mathbf{Z} \mathbf{H}_{\text{BP}, \alpha} = \text{GBP}(\alpha). \quad (11)$$

Moreover,

$$\|\mathbf{H}_{\text{BP}, \alpha}\|_\infty^\infty \leq n^2 \max_\beta \|\text{BP}(\beta)\|_\infty^\infty \sum_{a=0}^{\ell-1} (t \|\mathbf{D}\|_\infty^\infty)^a. \quad (12)$$

The public normalization algorithm computes the roots in Equation (6) for the n^2 scalar programs $\mathbf{e}_r^\top \text{BP} \mathbf{e}_s$ in parallel, evaluating every matrix product by a balanced tree. For every $(r, s) \in [n]^2$, it places

$$\mathbf{G}^{-1} \left(\mathbf{G}(\mathbf{h}_{\mathbf{e}_r^\top \text{BP} \mathbf{e}_s}^\top \otimes \mathbf{I}_n) \right)$$

in the (r, s) block of $\mathbf{M}_{\text{BP}}$. This is precisely the block arrangement paired with $\mathbf{R}_{\text{Can}}$ in Equation (9). Thus, when the public parameters, matrix dimensions, and entry bit lengths are polynomial in a total description length N , this algorithm has logspace-uniform, polynomial-size, bounded-fan-in Boolean circuits of depth

$$\log(\ell + 1) \log^{O(1)} N.$$

The following theorem [AMR26a] summarizes the homomorphic algebra needed in this work.

Theorem 3.4 (Homomorphic evaluation). *Let $h : \{0, 1\}^\ell \rightarrow \mathbb{Z}_q$ be computed by a depth- δ arithmetic circuit over $\mathbb{Z}_q$ with constants, addition, multiplication, and multiplication by constants. Suppose every integer wire value on a Boolean input has magnitude at most $B \geq 1$. There are deterministic algorithms*

- $\text{BPEvalM}(\text{pk}, h) \mapsto \mathbf{M}_h \in \{0, 1\}^{n \times mn^4}$,
- $\text{BPEvalH}(\text{pk}, h, \alpha) \mapsto \mathbf{H}_{h,\alpha} \in \mathbb{Z}_q^{nt^2 \times n}$

such that

$$\mathbf{G}\mathbf{M}_h\mathbf{P}_\alpha + \mathbf{Z}\mathbf{H}_{h,\alpha} = h(\alpha)\mathbf{G}. \quad (13)$$

The branching program $\text{Can}(\mathbf{M}_h)$ has width at most mn^4 and transition norm at most tmn^4 , and

$$\|\mathbf{H}_{h,\alpha}\|_\infty^\infty \leq ((tmn^4)^\ell + B)^\delta \left(1 + n^2(n(tmn^4)^2)^\ell\right) \sum_{a=0}^{\ell-1} (t\|\mathbf{D}\|_\infty^a)^a. \quad (14)$$

The first algorithm runs in time polynomial in λ, ℓ , and $|h|$; the second also has polynomial dependence on the time needed to evaluate $h(\alpha)$. If the public parameters, matrix dimensions, entry bit lengths, and the description of h have total length N , the first algorithm has logspace-uniform, polynomial-size, bounded-fan-in Boolean circuits of depth $(\delta + 1) \log^{O(1)} N$.

We will also need a *multi-hop variant* of the above theorem, which allows us to evaluate functions on intermediate points of the computation. This was also proven in [AMR26a]. Below ℓ denotes the length of α , whereas a denotes the number of coordinates of f .

Theorem 3.5 (Multi-Hop Evaluation). *Let $f = (f_0, \dots, f_{a-1})$ be represented by normalized programs and matrices satisfying Equation (13), and let $h : \{0, 1\}^\ell \times \mathbb{Z}_q^a \rightarrow \mathbb{Z}_q$ be as in Theorem 3.4. The wire bound B is required on the values occurring when h is evaluated at $(\alpha, f(\alpha))$. There are deterministic algorithms with interfaces*

- $\text{BPEvalM}(\text{pk}, h, \{\text{BP}_{f_i}\}_{i \in [a]}) \mapsto \mathbf{M}_{h \circ f}$,
- $\text{BPEvalH}(\text{pk}, h, \alpha, f(\alpha), \{\text{BP}_{f_i}\}_{i \in [a]}) \mapsto \{\mathbf{H}^{(i)}\}_{i \in [a+1]}$

that satisfy

$$\mathbf{G}\mathbf{M}_{h \circ f}\mathbf{P}_\alpha - h(\alpha, f(\alpha))\mathbf{G} = \sum_{i \in [a]} (\mathbf{G}\text{BP}_{f_i}(\alpha) - f_i(\alpha)\mathbf{G}) \mathbf{H}^{(i)} - \mathbf{Z}\mathbf{H}^{(a)}, \quad (15)$$

with

$$\|\mathbf{H}^{(i)}\|_\infty^\infty \leq ((tmn^4)^\ell + B)^\delta \quad (i \in [a]), \quad (16)$$

$$\|\mathbf{H}^{(a)}\|_\infty^\infty \leq ((tmn^4)^\ell + B)^\delta \cdot n^2 \left((n(tmn^4)^2)^\ell + \sum_{i \in [a]} \max_{\beta} \|\text{BP}_{f_i}(\beta)\|_\infty^\infty \right) \sum_{b=0}^{\ell-1} (t\|\mathbf{D}\|_\infty^b)^b. \quad (17)$$

The running-time statements of Theorem 3.4 continue to hold. Under the total-description-length convention in that theorem, the first algorithm has depth $(\delta + 1) \log^{O(1)} N$ and polynomial size.

When all public parameters and B are bounded by a fixed polynomial in λ and σ , Equation (14) and Equation (19) are $(\lambda\sigma)^{O(\ell\delta+\ell)}$. We retain the explicit formula whenever the polynomial-boundedness event matters.

Composing Theorem 3.4 with Theorem 3.5 when $\text{BP}_{f_i}(\alpha) = \mathbf{M}_{f_i}\mathbf{P}_\alpha$, $\mathbf{M}_{f_i} = \text{BPEvalM}(\text{pk}, f_i)$ establishes that

$$\mathbf{G}\mathbf{M}_{h \circ f}\mathbf{P}_\alpha + \mathbf{Z}\mathbf{H}_{h \circ f,\alpha} = h(\alpha, f(\alpha))\mathbf{G}, \quad \text{where} \quad \mathbf{H}_{h \circ f,\alpha} := \mathbf{H}^{(a)} + \sum_{i \in [a]} \mathbf{H}_{f_i,\alpha} \cdot \mathbf{H}^{(i)}, \quad (18)$$

and

$$\|\mathbf{H}_{h \circ f, \alpha}\|_{\infty}^{\infty} \leq ((tmn^4)^{\ell} + B)^{\delta} \left(\max_{i \in [a]} \|\mathbf{H}_{f_i, \alpha}\|_{\infty}^{\infty} + n^2 (n(tm n^4)^2)^{\ell} \sum_{b=0}^{\ell-1} (t \|\mathbf{D}\|_{\infty}^{\infty})^b \right). \quad (19)$$

Fix public matrices $\mathbf{U}_b \in \mathbb{Z}_q^{k \times r_b}$ for $b \in [c]$. Suppose that

$$\mathbf{G} \text{Can}(\mathbf{M}_{f_i})(\alpha) + \mathbf{Z} \mathbf{H}_{f_i, \alpha} + \sum_{b \in [c]} \mathbf{U}_b \mathbf{K}_{f_i, b, \alpha} = f_i(\alpha) \mathbf{G}. \quad (20)$$

If each coordinate f_i satisfies Equation (20), then set

$$\mathbf{K}_{h \circ f, b, \alpha} := \sum_{i \in [a]} \mathbf{K}_{f_i, b, \alpha} \mathbf{H}^{(i)}. \quad (21)$$

Substituting Equation (20) into Equation (15), with $\mathbf{H}_{h \circ f, \alpha}$ defined as in Equation (18), gives

$$\mathbf{G} \mathbf{M}_{h \circ f} \mathbf{P}_{\alpha} + \mathbf{Z} \mathbf{H}_{h \circ f, \alpha} + \sum_{b \in [c]} \mathbf{U}_b \mathbf{K}_{h \circ f, b, \alpha} = h(\alpha, f(\alpha)) \mathbf{G}.$$

The norm calculation proving Equation (19) applies verbatim to $\mathbf{K}_{h \circ f, b, \alpha}$, with $\max_i \|\mathbf{H}_{f_i, \alpha}\|_{\infty}^{\infty}$ replaced by $\max_i \|\mathbf{K}_{f_i, b, \alpha}\|_{\infty}^{\infty}$.

3.2 Homomorphic Read

Let $f : \{0, 1\}^{\ell} \rightarrow \{0, 1\}^{\ell}$ and $g : \{0, 1\}^{\ell} \rightarrow \mathbb{Z}_q$. Write $f_i(\alpha) = f(\alpha)_i$, and suppose that normalized programs and matrices satisfying Equation (13) have been computed for all f_i and for g . Absorb the input-independent initial and final matrices of $\text{Can}(\mathbf{M}_g)$ into its first and last transitions, and write the resulting transitions as

$$\text{MBP}_g(\beta) = \prod_{i=0}^{\ell-1} \mathbf{M}_{i, \beta_i},$$

where the paired matrices at every position have the same dimensions. Define

$$\text{Read}_g(x_0, \dots, x_{\ell-1}) := \prod_{i=0}^{\ell-1} (\mathbf{M}_{i,0} + x_i (\mathbf{M}_{i,1} - \mathbf{M}_{i,0})), \quad (22)$$

with the product evaluated by a balanced binary tree. Since every $f_i(\alpha)$ is Boolean,

$$\text{Read}_g(f_0(\alpha), \dots, f_{\ell-1}(\alpha)) = \text{MBP}_g(f(\alpha)). \quad (23)$$

For $r, s \in [n]$, let $\text{Read}_{g,r,s}(x) := \mathbf{e}_r^{\top} \text{Read}_g(x) \mathbf{e}_s$.

Lemma 3.6 (Homomorphic read). *There are deterministic polynomial-time algorithms*

- $\text{BPreadM}(\text{pk}, \{\mathbf{M}_{f_i}\}_{i \in [\ell]}, \mathbf{M}_g) \mapsto \mathbf{M}_{g \circ f} \in \{0, 1\}^{n \times mn^4},$
- $\text{BPreadH}(\text{pk}, \alpha, f(\alpha), \{\mathbf{M}_{f_i}, \mathbf{H}_{f_i, \alpha}\}_{i \in [\ell]}, \mathbf{M}_g, \mathbf{H}_{g, f(\alpha)}) \mapsto \mathbf{H}_{g \circ f, \alpha} \in \mathbb{Z}_q^{nt^2 \times n}$

such that

$$\mathbf{GCan}(\mathbf{M}_{g \circ f})(\alpha) + \mathbf{Z}\mathbf{H}_{g \circ f, \alpha} = g(f(\alpha))\mathbf{G}. \quad (24)$$

Under the polynomial parameter bounds stated after [Theorem 3.4](#), there is an absolute constant C such that

$$\|\mathbf{H}_{g \circ f, \alpha}\|_{\infty}^{\infty} \leq (\lambda\sigma)^{C\ell \log(\ell+1)} \left(1 + \max_{i \in [\ell]} \|\mathbf{H}_{f_i, \alpha}\|_{\infty}^{\infty} \right) + \|\mathbf{H}_{g, f(\alpha)}\|_{\infty}^{\infty}. \quad (25)$$

If the input programs satisfy [Equation \(20\)](#) for fixed matrices $\{\mathbf{U}_a\}_{a \in [c]}$, then $\mathbf{BPReadH}$ additionally receives $\mathbf{K}_{f_i, a, \alpha}$ and $\mathbf{K}_{g, a, f(\alpha)}$ and returns $\mathbf{K}_{g \circ f, a, \alpha} \in \mathbb{Z}_q^{r_a \times n}$ such that

$$\mathbf{GCan}(\mathbf{M}_{g \circ f})(\alpha) + \mathbf{Z}\mathbf{H}_{g \circ f, \alpha} + \sum_{a \in [c]} \mathbf{U}_a \mathbf{K}_{g \circ f, a, \alpha} = g(f(\alpha))\mathbf{G}. \quad (26)$$

For every $a \in [c]$,

$$\|\mathbf{K}_{g \circ f, a, \alpha}\|_{\infty}^{\infty} \leq (\lambda\sigma)^{C\ell \log(\ell+1)} \left(1 + \max_{i \in [\ell]} \|\mathbf{K}_{f_i, a, \alpha}\|_{\infty}^{\infty} \right) + \|\mathbf{K}_{g, a, f(\alpha)}\|_{\infty}^{\infty}. \quad (27)$$

If the public parameters and all matrix descriptions have total length N , $\mathbf{BPReadM}$ has logspace-uniform, polynomial-size, bounded-fan-in Boolean circuits of depth $\log(\ell+1) \log^{O(1)} N$.

Proof. Start with the affine matrix factors in [Equation \(22\)](#), evaluated on the normalized encodings of $f_0, \dots, f_{\ell-1}$. Evaluate the product in [Equation \(22\)](#) by the stated balanced binary tree. At each node, every entry of the product of its two child matrices has the form

$$(\mathbf{M}'\mathbf{M}'')_{r,s} = \sum_j M'_{r,j} M''_{j,s},$$

with at most mn^4 summands by [Theorem 3.4](#). Apply the multiplication operation of [Theorem 3.5](#) to every summand. In the affine case, at every such application propagate the coefficient of each $\mathbf{U}_a$ using [Equation \(21\)](#). To add the resulting branching programs without normalizing after each addition, horizontally concatenate their first transitions, use their block-diagonal concatenations at every intermediate transition, and vertically concatenate their last transitions. The value of the resulting program is exactly the displayed sum. Apply [Lemma 3.3](#) once, after all its summands have been included.

After the last level, for every $r, s \in [n]$ this gives a normalized matrix-output branching program $\mathbf{MBP}_{r,s}$ and a correction $\mathbf{H}_{r,s,\alpha}$ satisfying

$$\mathbf{GMBP}_{r,s}(\alpha) + \mathbf{Z}\mathbf{H}_{r,s,\alpha} = \mathbf{Read}_{g,r,s}(f(\alpha))\mathbf{G}. \quad (28)$$

In the affine case the same construction also gives matrices $\mathbf{K}_{r,s,a,\alpha}$ satisfying

$$\mathbf{GMBP}_{r,s}(\alpha) + \mathbf{Z}\mathbf{H}_{r,s,\alpha} + \sum_{a \in [c]} \mathbf{U}_a \mathbf{K}_{r,s,a,\alpha} = \mathbf{Read}_{g,r,s}(f(\alpha))\mathbf{G}.$$

Using the same horizontal, block-diagonal, and vertical concatenations, form the matrix-output branching program

$$\widehat{\mathbf{MBP}}(\alpha) := \sum_{r,s \in [n]} \mathbf{MBP}_{r,s}(\alpha) \mathbf{e}_r \mathbf{e}_s^{\top}.$$

Summing Equation (28) after right multiplication by $\mathbf{e}_r \mathbf{e}_s^\top$ gives

$$\widehat{\mathbf{GMBP}}(\alpha) + \mathbf{Z} \sum_{r,s \in [n]} \mathbf{H}_{r,s,\alpha} \mathbf{e}_r \mathbf{e}_s^\top = \mathbf{GRead}_g(f(\alpha)) = \mathbf{GMBP}_g(f(\alpha)), \quad (29)$$

where the last equality is Equation (23).

Normalize $\widehat{\mathbf{MBP}}$ using Lemma 3.3; call its correction $\tilde{\mathbf{H}}_\alpha$. Define

$$\mathbf{H}_{g \circ f, \alpha} := \tilde{\mathbf{H}}_\alpha + \sum_{r,s \in [n]} \mathbf{H}_{r,s,\alpha} \mathbf{e}_r \mathbf{e}_s^\top + \mathbf{H}_{g,f(\alpha)}. \quad (30)$$

Combining Equation (29), normalization, and the invariant for g at $f(\alpha)$ proves Equation (24).

We next prove the norm bound. By Theorem 3.4, every transition of a normalized input program has norm at most tmn^4 ; hence its value has norm at most $(tmn^4)^\ell$. The same bound holds for every matrix occurring in the balanced evaluation of Equation (22), because each is a product of at most ℓ matrices of norm at most tmn^4 . The norm calculation underlying Equation (19) therefore multiplies an incoming matrix norm by at most $2(tm n^4)^\ell$. There are at most mn^4 summands in each entry. Before normalization their program value has norm at most $mn^4(tm n^4)^{2\ell}$, so Equation (12) adds at most

$$n^2 mn^4 (tm n^4)^{2\ell} \sum_{a=0}^{\ell-1} (t \|\mathbf{D}\|_\infty)^a. \quad (31)$$

Consequently, at each level of the balanced evaluation, the largest incoming matrix norm is multiplied by at most $2mn^4(tm n^4)^\ell$, and the quantity in Equation (31) is added. Under the polynomial parameter bounds, both are at most $(\lambda\sigma)^{C\ell}$ after increasing the absolute constant C . The affine factors at the leaves satisfy the same bound with initial value $1 + \max_i \|\mathbf{H}_{f_i, \alpha}\|_\infty$. Since the balanced evaluation has $\lceil \log_2 \ell \rceil$ levels, its entry matrices satisfy the first term of Equation (25). Combining the n^2 entries and normalizing once more costs another $(\lambda\sigma)^{O(\ell)}$ factor, which is absorbed by increasing C ; Equation (30) then adds $\mathbf{H}_{g,f(\alpha)}$.

For the affine assertion, use Equation (21) at every invocation of Theorem 3.5. Normalization adds no coefficient of $\mathbf{U}_a$. Thus, after assembling the entries, set

$$\mathbf{K}_{g \circ f, a, \alpha} := \sum_{r,s \in [n]} \mathbf{K}_{r,s,a,\alpha} \mathbf{e}_r \mathbf{e}_s^\top + \mathbf{K}_{g,a,f(\alpha)}.$$

Substitution proves Equation (26), and the same level-by-level bound proves Equation (27).

Finally, the balanced product has $O(\log(\ell+1))$ levels. At each level, all entries and all summands are processed in parallel, and the normalization procedure following Lemma 3.3 has polylogarithmic depth. This proves the stated depth and uniformity of $\mathbf{BReadM}$. $\square$

3.3 The New Key Equation

For $f : \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$, define $f^{(0)} = \text{id}$ and $f^{(T+1)} = f \circ f^{(T)}$, and write $f_j^{(T)}(\alpha) = f^{(T)}(\alpha)_j$.

Theorem 3.7 (Iterated evaluation). *Let $f : \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$ be computed by a depth- δ polynomially bounded arithmetic circuit. For every power of two $T \geq 1$, there are deterministic algorithms*

- $\text{EvalM}(\text{pk}, f, T) \mapsto \{\mathbf{M}_{f,T,j} \in \{0, 1\}^{n \times mn^4}\}_{j \in [\ell]},$

- $\text{EvalH}(\text{pk}, f, \alpha, 1^T) \mapsto \{\mathbf{H}_{f,T,j,\alpha} \in \mathbb{Z}_q^{nt^2 \times n}\}_{j \in [\ell]},$
- $\text{ExpandP}(\text{pk}, \alpha) \mapsto \mathbf{P}_\alpha \in \mathbb{Z}_q^{mn^4 \times n}$

such that, for every $j \in [\ell]$ and $\alpha \in \{0, 1\}^\ell$,

$$\mathbf{G}\mathbf{M}_{f,T,j}\mathbf{P}_\alpha + \mathbf{Z}\mathbf{H}_{f,T,j,\alpha} = f_j^{(T)}(\alpha)\mathbf{G}. \quad (32)$$

Moreover,

$$\|\mathbf{H}_{f,T,j,\alpha}\|_\infty \leq (\lambda\sigma)^{O(\ell\delta + \ell\log(\ell+1)\log(T+1))}, \quad (33)$$

$$\|\mathbf{P}_\alpha\|_\infty \leq t^\ell. \quad (34)$$

The integer T is given in binary to EvalM and in unary to EvalH . The algorithm EvalM runs in time $\text{poly}(\lambda, |f|, \ell, \log(T+1))$, while EvalH runs in time $T \cdot \text{poly}(\lambda, |f|, \ell, \log(T+1))$.

Proof. For $T = 1$, apply [Theorem 3.4](#) to every coordinate f_j . Suppose programs have been constructed for all coordinates of $f^{(2^s)}$. For every j , apply [Lemma 3.6](#) with the vector function $f^{(2^s)}$ as the encoded address and $f_j^{(2^s)}$ as the encoded table. The output represents

$$f_j^{(2^s)}(f^{(2^s)}(\alpha)) = f_j^{(2^{s+1})}(\alpha).$$

For $T = 2^s$, this constructs $f^{(T)}$ using exactly s homomorphic-read layers.

To justify the uniform norm bound, let $E_s := \max_{\alpha,j} \|\mathbf{H}_{f,2^s,j,\alpha}\|_\infty$. The read at the point $\beta = f^{(2^s)}(\alpha)$ gives the uniform recurrence

$$E_{s+1} \leq (\lambda\sigma)^{C\ell\log(\ell+1)}(1 + E_s) + E_s$$

where C is from [Equation \(25\)](#). Since $E_0 \leq (\lambda\sigma)^{O(\ell\delta)}$ and $s = \log_2 T$, unrolling this recurrence proves [Equation \(33\)](#). The public algorithm manipulates only normalized branching programs of fixed polynomial width and performs $\log_2 T$ squarings, so its running time is polynomial in $\log(T+1)$. The recursive evaluation used by EvalH has $O(T)$ nodes, each of which performs a polynomial amount of work and passes the value of the corresponding iterate to its parent. Finally, [Equation \(34\)](#) is [Equation \(10\)](#). $\square$

Horizontally concatenate the coordinate matrices and corrections:

$$\mathbf{M}_{f,T} := [\mathbf{M}_{f,T,0} \mid \cdots \mid \mathbf{M}_{f,T,\ell-1}], \quad \widehat{\mathbf{H}}_{f,T,\alpha} := -[\mathbf{H}_{f,T,0,\alpha} \mid \cdots \mid \mathbf{H}_{f,T,\ell-1,\alpha}],$$

and put $\tilde{\mathbf{P}}_\alpha = \mathbf{I}_\ell \otimes \mathbf{P}_\alpha$. These matrices have dimensions

$$\mathbf{M}_{f,T} \in \mathbb{Z}_q^{n \times \ell mn^4}, \quad \tilde{\mathbf{P}}_\alpha \in \mathbb{Z}_q^{\ell mn^4 \times \ell n}, \quad \widehat{\mathbf{H}}_{f,T,\alpha} \in \mathbb{Z}_q^{nt^2 \times \ell n},$$

and satisfy

$$\|\widehat{\mathbf{H}}_{f,T,\alpha}\|_\infty \leq \ell \max_j \|\mathbf{H}_{f,T,j,\alpha}\|_\infty, \quad \|\tilde{\mathbf{P}}_\alpha\|_\infty \leq t^\ell.$$

The coordinate identities in [Equation \(32\)](#) give the final key equation

$$\mathbf{G} \left(\mathbf{M}_{f,T} \tilde{\mathbf{P}}_\alpha - f^{(T)}(\alpha)^\top \otimes \mathbf{I}_n \right) = \mathbf{Z} \widehat{\mathbf{H}}_{f,T,\alpha}. \quad (35)$$

4 Average-Case Time and Space Hardness

We now construct two correlation-intractable hash families. The first is compact in the running time of a space-bounded computation; the second is compact in the width of a bounded-depth circuit. Both use the affine form of the branching-program invariant and the following diagonalization lemma.

Lemma 4.1 (Diagonalization). *Let $\mathcal{F} = (\mathcal{F}_\lambda)_\lambda$ be a family of functions $F : \mathcal{X}_\lambda \rightarrow \mathcal{Y}_\lambda$. Suppose $(\text{KeyGen}, \text{CIHash})$ is (s, ε) -correlation intractable for $\mathcal{R}_F$ for every ensemble $F = (F_\lambda)_\lambda$ with $F_\lambda \in \mathcal{F}_\lambda$, and suppose $\text{supp}(\text{KeyGen}(1^\lambda)) \subseteq \mathcal{X}_\lambda$. Assume also that $s(\lambda)$ includes the $O(|K|)$ time needed to copy a key. Define*

$$g_\lambda(k) := \text{CIHash}(k, k).$$

Then every such ensemble F satisfies

$$\Pr_{K \xleftarrow{\$} \text{KeyGen}(1^\lambda)} [F_\lambda(K) = g_\lambda(K)] \leq \varepsilon(\lambda). \quad (36)$$

Proof. If Equation (36) fails, the correlation-finding adversary outputs $x = K$ on input the hash key K . Its hash value is $\text{CIHash}(K, K) = g_\lambda(K) = F_\lambda(K)$ with probability greater than ε , contradicting correlation intractability for $\mathcal{R}_F$. $\square$

Both constructions produce a scalar table whose values at n designated addresses are the n output bits. If these addresses are $\alpha_0, \dots, \alpha_{n-1}$, define

$$\mathbf{v} := \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_{n-1} \end{bmatrix} \in \{0, 1\}^{n^2}. \quad (37)$$

For any $\mathbf{M} \in \mathbb{Z}_q^{n \times mn^4}$,

$$\mathbf{M}[\mathbf{P}_{\alpha_0} \mid \dots \mid \mathbf{P}_{\alpha_{n-1}}] \mathbf{v} = \sum_{j=0}^{n-1} \mathbf{M} \mathbf{P}_{\alpha_j} \mathbf{e}_j.$$

Thus coordinate equations with right-hand sides $y_j \mathbf{G}$ combine into the single right-hand side $\mathbf{G} \mathbf{y}$.

4.1 Space Hardness

We first state the RAM consequence of Theorem 3.7. A read-only memory is a scalar table represented by a normalized matrix branching program. Every random-access query is implemented by Lemma 3.6; all remaining coordinates of a RAM transition are evaluated by Theorem 3.5.

Corollary 4.2 (Iterated RAM evaluation). *Let C_x be the transition-circuit template of a deterministic random-access program with d -bit configurations, read-only input fixed to x , and read-only program memory $p : \{0, 1\}^d \rightarrow \{0, 1\}$. Assume that p is computed by a depth- $O(d)$, polynomial-size arithmetic circuit whose wire values have magnitude at most $(\lambda\sigma)^{O(1)}$. Assume also that each transition makes a constant number of memory reads at addresses in its input configuration and that, after replacing the read gates by input wires, C_x has depth $O(\log d)$ and polynomial size.*

Set $\mathbf{M}_p := \text{BPEvalM}(\text{pk}, p)$, and fix $\mathbf{W} \in \{0, 1\}^{n \times mn^4}$ and $\mathbf{R} \in \mathbb{Z}_q^{k \times mn^4}$ such that $\mathbf{G} \mathbf{W} = \mathbf{G} \mathbf{M}_p + \mathbf{R}$. There are deterministic algorithms such that, for every power of two $T \geq 1$,

- $\text{RAMM}(\text{pk}, d, C_x, \mathbf{W}, T) \mapsto \mathbf{M}_{x,T} \in \{0,1\}^{n \times mn^4},$
- $\text{RAMH}(\text{pk}, d, C_x, p, \mathbf{W}, \alpha, 1^T) \mapsto (\mathbf{H}_{x,T,\alpha}, \mathbf{K}_{x,T,\alpha}) \in \mathbb{Z}_q^{nt^2 \times n} \times \mathbb{Z}_q^{mn^4 \times n}.$

For every $\alpha \in \{0,1\}^d$, their outputs satisfy

$$\mathbf{G}\mathbf{M}_{x,T}\mathbf{P}_\alpha + \mathbf{Z}\mathbf{H}_{x,T,\alpha} + \mathbf{R}\mathbf{K}_{x,T,\alpha} = \text{out}\left((C_x^p)^{(T)}(\alpha)\right) \mathbf{G}. \quad (38)$$

They also satisfy

$$\max\{\|\mathbf{H}_{x,T,\alpha}\|_\infty, \|\mathbf{K}_{x,T,\alpha}\|_\infty\} \leq (\lambda\sigma)^{O(d^2 + (1+\log T)d\log(d+1))}. \quad (39)$$

The running times of RAMM and RAMH are, respectively, $\text{poly}(\lambda, d, |C_x|, |x|, \log(T+1))$ and $T \cdot \text{poly}(\lambda, d, |C_x|, |x|, |p|, \log(T+1))$.

Proof. At a program-memory read from address β , RAMH computes $\mathbf{H}_{p,\beta} := \text{BPEvalH}(\text{pk}, p, \beta)$. By Equation (13) and the assumed identity for $\mathbf{W}$,

$$\mathbf{G}\mathbf{W}\mathbf{P}_\beta + \mathbf{Z}\mathbf{H}_{p,\beta} - \mathbf{R}\mathbf{P}_\beta = p(\beta)\mathbf{G}.$$

Thus every program-memory read satisfies Equation (20), with $\mathbf{R}$ as the affine matrix and $-\mathbf{P}_\beta$ as its coefficient. Compile one transition coordinate by coordinate using Theorems 3.4 and 3.5 and Lemma 3.6, propagating the coefficient of $\mathbf{R}$ using Equation (21), and then apply Theorem 3.7.

By Equations (10) and (14),

$$\max\{\|\mathbf{H}_{p,\beta}\|_\infty, \|\mathbf{P}_\beta\|_\infty\} \leq (\lambda\sigma)^{O(d^2)}.$$

By Equations (19), (25) and (27), one transition contributes $(\lambda\sigma)^{O(d\log(d+1))}$ to the common recurrence for the matrices multiplying $\mathbf{Z}$ and $\mathbf{R}$. The direct encoding of x has the same base-level bound: Its table lookup has a depth- $O(d)$ arithmetic circuit with Boolean wire values. There are $\log_2 T$ squaring layers, which proves Equation (39). Finally, RAMH computes $\mathbf{H}_{p,\beta}$ at the constant number of program-memory reads in each transition, so the claimed running times follow from Theorem 3.7. $\square$

Construction. Fix $d, \kappa \geq 1$. Masks are indexed by $\{0, \dots, \kappa\}$; including the endpoint permits computations lasting 2^κ steps.

KeyGen_{sp}($1^\lambda, 1^d, 1^\kappa$): Sample $\text{pk} \xleftarrow{\$} \text{Setup}(1^\lambda, 1^t)$,

$$\mathbf{W} \xleftarrow{\$} \{0,1\}^{n \times mn^4}, \quad \mathbf{U}_i \xleftarrow{\$} \{0,1\}^{n \times n} \quad (i = 0, \dots, \kappa), \quad \mathbf{a} \xleftarrow{\$} \{0,1\}^n.$$

Return $\text{hk} = (\text{pk}, \mathbf{W}, (\mathbf{U}_i)_{i=0}^\kappa, \mathbf{a})$.

CIHash_{sp}(hk, x): Let C_x be the universal d -configuration transition circuit with read-only input x and read-only program memory. For $i = 0, \dots, \kappa$, compute

$$\mathbf{M}_i := \text{RAMM}(\text{pk}, d, C_x, \mathbf{W}, 2^i), \quad \mathbf{M} := \sum_{i=0}^{\kappa} \mathbf{U}_i \mathbf{M}_i.$$

Let $\alpha_j \in \{0,1\}^d$ be the standard initial configuration selecting the j -th program in the read-only memory. Return

$$\mathbf{G}^{-1}(\mathbf{G}\mathbf{M}[\mathbf{P}_{\alpha_0} \mid \dots \mid \mathbf{P}_{\alpha_{n-1}}]\mathbf{v} + \mathbf{G}\mathbf{a}) \in \{0,1\}^n. \quad (40)$$

For a uniformly generated ensemble of deterministic programs $\mathcal{M}_{\lambda,0}, \dots, \mathcal{M}_{\lambda,n-1}$ that halt within T steps and whose halting configurations are fixed by their transition maps, fix an input length $L_\lambda \leq 2^{O(d)}$ and define

$$\mathcal{R}_{\mathcal{M},T} := \left\{ (x, y) : x \in \{0, 1\}^{L_\lambda}, y_j = \text{out}(\mathcal{M}_{\lambda,j}^{(T)}(x)) \text{ for every } j \in [n] \right\}. \quad (41)$$

Theorem 4.3 (Correlation intractability for space-bounded computation). *Let d, κ be polynomially bounded, let $1 \leq T \leq 2^\kappa$, and let the uniformly generated programs in Equation (41) compile to d -bit configurations in the fixed model of Section 2. Put*

$$r_{\text{sp}} := mn^4 + (\kappa + 1)n + 1, \quad (42)$$

$$\beta_{\text{sp}} := (\lambda\sigma)^{C(d^2 + (\kappa+1)d \log(d+1))} \quad (43)$$

for a sufficiently large absolute constant C , and assume $\beta_{\text{sp}} < q$.

If $(k, m, t, q, \sigma, r_{\text{sp}}, \beta_{\text{sp}})$ -ExtDSIS has success probability at most ε for algorithms of running time

$$\text{poly}(s, T, \lambda, d, \kappa, |\mathcal{M}_{\lambda,0}| + \dots + |\mathcal{M}_{\lambda,n-1}|),$$

then $(\text{KeyGen}_{\text{sp}}, \text{CIHash}_{\text{sp}})$ is $(s, \varepsilon + \delta_{\text{tail}})$ -correlation intractable for $\mathcal{R}_{\mathcal{M},T}$.

Proof. Suppose an adversary finds a correlation with probability greater than $\varepsilon + \delta_{\text{tail}}$. Conditioned on the Gaussian tail event, its success probability is greater than ε . We construct an extended-DSIS solver. Partition its auxiliary uniform matrix as

$$\mathbf{U} = [\mathbf{R} \mid \widehat{\mathbf{U}}_0 \mid \dots \mid \widehat{\mathbf{U}}_\kappa \mid \mathbf{u}_*],$$

where $\mathbf{R}$ has mn^4 columns, every $\widehat{\mathbf{U}}_i$ has n columns, and $\mathbf{u}_*$ has one column.

Let $p : \{0, 1\}^d \rightarrow \{0, 1\}$ contain the concatenated descriptions of $\mathcal{M}_{\lambda,0}, \dots, \mathcal{M}_{\lambda,n-1}$, followed by zeros. On Boolean inputs it is computed by

$$p(\alpha) = \sum_{\substack{\beta \in \{0,1\}^d \\ p(\beta)=1}} \prod_{i=0}^{d-1} (\beta_i \alpha_i + (1 - \beta_i)(1 - \alpha_i)).$$

Balanced products and a balanced sum give depth $O(d)$ and size $O\left(d \sum_{j=0}^{n-1} |\mathcal{M}_{\lambda,j}|\right)$; all wire values on Boolean inputs lie in $\{0, 1\}$. Set

$$\mathbf{M}_p := \text{BPEvalM}(\text{pk}, p).$$

Program

$$\mathbf{W} := \mathbf{G}^{-1}(\mathbf{G}\mathbf{M}_p + \mathbf{R}). \quad (44)$$

By Equation (2), these matrices satisfy $\mathbf{G}\mathbf{W} = \mathbf{G}\mathbf{M}_p + \mathbf{R}$. Let $h = \lceil \log_2 T \rceil \leq \kappa$ and program

$$\mathbf{U}_i := \begin{cases} \mathbf{G}^{-1}(\widehat{\mathbf{U}}_i), & i \neq h, \\ \mathbf{G}^{-1}(\widehat{\mathbf{U}}_h + \mathbf{G}), & i = h, \end{cases} \quad \mathbf{a} := \mathbf{G}^{-1}(\mathbf{u}_*). \quad (45)$$

Every programmed object has exactly the distribution used by $\text{KeyGen}_{\text{sp}}$, by Equation (2). Give this key to the adversary and let x be its output.

For every $j \in [n]$, compute

$$(\mathbf{H}_{h,j}, \mathbf{K}_{h,j}) := \text{RAMH}(\text{pk}, d, C_x, p, \mathbf{W}, \alpha_j, 1^{2^h}).$$

By Equation (38), these matrices satisfy

$$\mathbf{G}\mathbf{M}_h\mathbf{P}_{\alpha_j} + \mathbf{Z}\mathbf{H}_{h,j} + \mathbf{R}\mathbf{K}_{h,j} = \text{out}\left((C_x^p)^{(2^h)}(\alpha_j)\right) \mathbf{G} = \text{out}(\mathcal{M}_{\lambda,j}^{(T)}(x)) \mathbf{G}. \quad (46)$$

The last equality holds because $2^h \geq T$ and every halting configuration is fixed by the transition map. Moreover, Equation (45) implies

$$\mathbf{G}\mathbf{U}_i = \begin{cases} \widehat{\mathbf{U}}_i, & i \neq h, \\ \widehat{\mathbf{U}}_h + \mathbf{G}, & i = h. \end{cases}$$

Substituting these identities and Equation (46) into the definition of $\mathbf{M}$ gives

$$\mathbf{G}\mathbf{M}\mathbf{P}_{\alpha_j} + \mathbf{Z}\mathbf{H}_{h,j} + \mathbf{R}\mathbf{K}_{h,j} - \sum_{i=0}^{\kappa} \widehat{\mathbf{U}}_i \mathbf{M}_i \mathbf{P}_{\alpha_j} = \text{out}(\mathcal{M}_{\lambda,j}^{(T)}(x)) \mathbf{G}. \quad (47)$$

Define

$$\mathbf{H} := \sum_{j=0}^{n-1} \mathbf{H}_{h,j} \mathbf{e}_j, \quad \mathbf{K} := \begin{bmatrix} \sum_{j=0}^{n-1} \mathbf{K}_{h,j} \mathbf{e}_j \\ - \sum_{j=0}^{n-1} \mathbf{M}_0 \mathbf{P}_{\alpha_j} \mathbf{e}_j \\ \vdots \\ - \sum_{j=0}^{n-1} \mathbf{M}_{\kappa} \mathbf{P}_{\alpha_j} \mathbf{e}_j \end{bmatrix}.$$

Combining Equation (47) over all j using Equation (37) yields

$$\mathbf{G}\mathbf{M}[\mathbf{P}_{\alpha_0} \mid \cdots \mid \mathbf{P}_{\alpha_{n-1}}] \mathbf{v} + \mathbf{Z}\mathbf{H} + [\mathbf{R} \mid \widehat{\mathbf{U}}_0 \mid \cdots \mid \widehat{\mathbf{U}}_{\kappa}] \mathbf{K} = \mathbf{G} \begin{bmatrix} \text{out}(\mathcal{M}_{\lambda,0}^{(T)}(x)) \\ \vdots \\ \text{out}(\mathcal{M}_{\lambda,n-1}^{(T)}(x)) \end{bmatrix}. \quad (48)$$

By Equation (39), $h \leq \kappa$, Equation (10), and $\|\mathbf{M}_i\|_{\infty}^{\infty} \leq mn^4$ for every i ,

$$\max\{\|\mathbf{H}\|_{\infty}, \|\mathbf{K}\|_{\infty}\} \leq \beta_{\text{sp}} \quad (49)$$

after increasing C if necessary.

If the adversary's output is a correlation, the vector on the right of Equation (48) equals $\text{CIHash}_{\text{sp}}(\text{hk}, x)$. Applying $\mathbf{G}$ to Equation (40) and subtracting the two identities gives

$$\mathbf{Z}\mathbf{H} + [\mathbf{R} \mid \widehat{\mathbf{U}}_0 \mid \cdots \mid \widehat{\mathbf{U}}_{\kappa}] \mathbf{K} - \mathbf{u}_{\star} = \mathbf{0}.$$

At this point $\mathbf{H} \in \mathbb{Z}_q^{nt^2}$ and $\mathbf{K} \in \mathbb{Z}_q^{r_{\text{sp}}-1}$. Using their centered integer representatives, set

$$\mathbf{w}_{\text{sp}} := \begin{bmatrix} \mathbf{H} \\ \mathbf{K} \\ -1 \end{bmatrix} \in \mathbb{Z}^{nt^2+r_{\text{sp}}}.$$

This is an extended-DSIS solution. It is nonzero because its last coordinate is -1 , and Equation (49) bounds its norm by β_{sp} . Computing $\mathbf{H}$ and $\mathbf{K}$ evaluates at $O(T)$ recursive nodes, each with the stated polynomial overhead, giving the claimed reduction time. $\square$

Corollary 4.4 (Average-case space hardness). *Let N be the key length. On every N -bit string K , define $g_{\lambda,d,\kappa}^{\text{sp}}(K) = \text{CHash}_{\text{sp}}(K, K)$ if K parses as a key, and define its value to be 0^n otherwise. Let*

$$L_{\text{sp}} := \{(1^\lambda, K, i) : i \in [n], g_{\lambda,d,\kappa}^{\text{sp}}(K)_i = 1\}.$$

Under the decomposed-SIS parameters of Theorem 4.3, every uniformly generated algorithm F whose n bit-output programs halt within some $T \leq 2^\kappa$ steps and whose compiled configurations have at most d bits satisfies

$$\Pr_{K \leftarrow \text{KeyGen}_{\text{sp}}(1^\lambda, 1^d, 1^\kappa)} [F(K) = g_{\lambda,d,\kappa}^{\text{sp}}(K)] \leq \varepsilon(\lambda) + \delta_{\text{tail}}(\lambda)$$

which is negligible.

Proof. Instantiate Equation (41) with $L_\lambda = N$ and with the batch of n bit-output programs for F , and apply Lemma 4.1 to Theorem 4.3. If a vector-output algorithm uses s bits of work space, its bit-output programs use $s + O(\log n + \log N)$ bits to hold the output index and read-only addresses; the statement therefore applies whenever these compiled configurations fit in d bits. $\square$

4.2 Time Hardness

We next encode a layered Boolean circuit entirely by scalar matrix branching programs. The parameter κ is the address length: when the circuit size and its input and output lengths are at most 2^κ , every input position, output position, and gate in a layer can be indexed by a string in $\{0, 1\}^\kappa$. After inserting copy gates, all wires go between consecutive layers. At layer i , the functions

$$\phi_{i,\text{L}}, \phi_{i,\text{R}} : \{0, 1\}^\kappa \rightarrow \{0, 1\}^\kappa$$

give the addresses of the left and right predecessors, and $\phi_{i,\text{N}} : \{0, 1\}^\kappa \rightarrow \{0, 1\}$ indicates whether the gate is NAND rather than a copy gate. For $b \in [\kappa]$, write

$$\phi_{i,\text{L},b}(\alpha) := \phi_{i,\text{L}}(\alpha)_b, \quad \phi_{i,\text{R},b}(\alpha) := \phi_{i,\text{R}}(\alpha)_b.$$

Construction. Fix a layer bound T . The index b on $\mathbf{W}_{i,\text{L},b}$ and $\mathbf{W}_{i,\text{R},b}$ denotes the b -th output coordinate of the corresponding predecessor address; these matrices are supplied in that order to BPReadM below.

$\text{KeyGen}_{\text{tm}}(1^\lambda, 1^\kappa, 1^T)$: Sample $\text{pk} \xleftarrow{\$} \text{Setup}(1^\lambda, 1^t)$ and independent uniform bit matrices

$$\begin{aligned} \mathbf{W}_{i,\text{L},b}, \mathbf{W}_{i,\text{R},b} &\xleftarrow{\$} \{0, 1\}^{n \times mn^4} & (i \in [T], b \in [\kappa]), \\ \mathbf{W}_{i,\text{N}} &\xleftarrow{\$} \{0, 1\}^{n \times mn^4} & (i \in [T]), \end{aligned}$$

together with $\mathbf{a} \xleftarrow{\$} \{0, 1\}^n$. Return all these values as hk .

$\text{CIHash}_{\text{tm}}(\text{hk}, x)$: Identify $\alpha \in \{0, 1\}^\kappa$ with the integer represented by its binary expansion, and define

$$X_x(\alpha) := \begin{cases} x_\alpha, & \alpha < |x|, \\ 0, & \alpha \geq |x|. \end{cases}$$

Equivalently, if j_b is the b -th bit of the κ -bit representation of j , then

$$X_x(\alpha) = \sum_{j \in [\min\{|x|, 2^\kappa\}]} x_j \prod_{b=0}^{\kappa-1} (j_b \alpha_b + (1 - j_b)(1 - \alpha_b)).$$

Compute

$$\mathbf{M}^{(0)} := \text{BPEvalM}(\text{pk}, X_x).$$

For $i = 0, \dots, T - 1$, compute

$$\begin{aligned} \mathbf{M}_L^{(i)} &:= \text{BPReadM}(\text{pk}, \{\mathbf{W}_{i,L,b}\}_{b \in [\kappa]}, \mathbf{M}^{(i)}), \\ \mathbf{M}_R^{(i)} &:= \text{BPReadM}(\text{pk}, \{\mathbf{W}_{i,R,b}\}_{b \in [\kappa]}, \mathbf{M}^{(i)}), \\ \mathbf{M}^{(i+1)} &:= \text{BPEvalM}(\text{pk}, \text{Gate}, \mathbf{W}_{i,N}, \mathbf{M}_L^{(i)}, \mathbf{M}_R^{(i)}), \end{aligned}$$

where

$$\text{Gate}(c, l, r) := (1 - c)l + c(1 - lr).$$

On Boolean inputs this equals $1 - lr$ when $c = 1$ and l when $c = 0$; in particular, it is computed by a constant-depth arithmetic circuit. If $\alpha_0, \dots, \alpha_{n-1}$ are the designated output-gate addresses, return

$$\mathbf{G}^{-1}(\mathbf{GM}^{(T)}[\mathbf{P}_{\alpha_0} \mid \dots \mid \mathbf{P}_{\alpha_{n-1}}] \mathbf{v} + \mathbf{Ga}). \quad (50)$$

Theorem 4.5 (Correlation intractability for bounded-depth circuits). *Let $C_\lambda : \{0, 1\}^{L_\lambda} \rightarrow \{0, 1\}^n$ be a logspace-uniform bounded-fan-in Boolean circuit whose layered NAND-and-copy conversion has size S and depth at most T , where $L_\lambda, n, S \leq 2^\kappa$. Put*

$$r_{\text{tm}} := T(2\kappa + 1)mn^4 + 1, \quad (51)$$

$$\beta_{\text{tm}} := (\lambda\sigma)^{C(\kappa^2 + T\kappa \log(\kappa+1))} \quad (52)$$

for a sufficiently large absolute constant C , and assume $\beta_{\text{tm}} < q$.

If $(k, m, t, q, \sigma, r_{\text{tm}}, \beta_{\text{tm}})$ -ExtDSIS has success probability at most ε for algorithms of running time

$$\text{poly}(s, S, \lambda, \kappa, T),$$

then $(\text{KeyGen}_{\text{tm}}, \text{CIHash}_{\text{tm}})$ is $(s, \varepsilon + \delta_{\text{tail}})$ -correlation intractable for

$$\mathcal{R}_C = \{(x, y) : x \in \{0, 1\}^{L_\lambda}, y = C_\lambda(x)\}.$$

Proof. Suppose a correlation adversary succeeds with probability greater than $\varepsilon + \delta_{\text{tail}}$. Conditioned on the Gaussian tail event, it succeeds with probability greater than ε . Take the layered NAND-and-copy conversion of C_λ and, if necessary, append copy layers so that there are exactly T layers.

Form its routing functions $\phi_{i,L}, \phi_{i,R}, \phi_{i,N}$. Partition the auxiliary matrix in the extended-DSIS instance as

$$\mathbf{U} = [\{\mathbf{U}_{i,L,b}, \mathbf{U}_{i,R,b}\}_{i,b} \mid \{\mathbf{U}_{i,N}\}_i \mid \mathbf{u}_\star],$$

where every block preceding the final one-column block $\mathbf{u}_\star$ has mn^4 columns.

Each scalar routing function is computed on Boolean inputs by the formula

$$\phi(\alpha) = \sum_{\substack{j \in [2^\kappa] \\ \phi(j)=1}} \prod_{b=0}^{\kappa-1} (j_b \alpha_b + (1 - j_b)(1 - \alpha_b)),$$

where unused gate positions have value zero. The sum has at most S terms, apart from the appended layers, which have at most n terms. Thus balanced products and a balanced sum give size $O(\kappa(S + n))$ and depth $O(\kappa)$. For every $\phi_{i,L,b}, \phi_{i,R,b}, \phi_{i,N}$, set

$$\mathbf{M}_\phi := \text{BPEvalM}(\text{pk}, \phi), \quad \mathbf{H}_{\phi,\alpha} := \text{BPEvalH}(\text{pk}, \phi, \alpha).$$

Then [Equations \(13\)](#) and [\(14\)](#) give

$$\mathbf{GM}_\phi \mathbf{P}_\alpha + \mathbf{ZH}_{\phi,\alpha} = \phi(\alpha) \mathbf{G}, \quad \|\mathbf{H}_{\phi,\alpha}\|_\infty^\infty \leq (\lambda\sigma)^{O(\kappa^2)}. \quad (53)$$

Program the corresponding hash-key matrices by

$$\begin{aligned} \mathbf{W}_{i,L,b} &:= \mathbf{G}^{-1}(\mathbf{GM}_{\phi_{i,L,b}} + \mathbf{U}_{i,L,b}), \\ \mathbf{W}_{i,R,b} &:= \mathbf{G}^{-1}(\mathbf{GM}_{\phi_{i,R,b}} + \mathbf{U}_{i,R,b}), \\ \mathbf{W}_{i,N} &:= \mathbf{G}^{-1}(\mathbf{GM}_{\phi_{i,N}} + \mathbf{U}_{i,N}), \quad \mathbf{a} := \mathbf{G}^{-1}(\mathbf{u}_\star). \end{aligned}$$

All programmed matrices are exactly uniform by [Equation \(2\)](#). Moreover, [Equation \(53\)](#) gives

$$\mathbf{GW}_{i,L,b} \mathbf{P}_\alpha + \mathbf{ZH}_{\phi_{i,L,b},\alpha} - \mathbf{U}_{i,L,b} \mathbf{P}_\alpha = \phi_{i,L,b}(\alpha) \mathbf{G}, \quad (54)$$

$$\mathbf{GW}_{i,R,b} \mathbf{P}_\alpha + \mathbf{ZH}_{\phi_{i,R,b},\alpha} - \mathbf{U}_{i,R,b} \mathbf{P}_\alpha = \phi_{i,R,b}(\alpha) \mathbf{G}, \quad (55)$$

$$\mathbf{GW}_{i,N} \mathbf{P}_\alpha + \mathbf{ZH}_{\phi_{i,N},\alpha} - \mathbf{U}_{i,N} \mathbf{P}_\alpha = \phi_{i,N}(\alpha) \mathbf{G}.$$

Give the resulting key to the adversary and let x be its output.

Set $\mathbf{H}_\alpha^{(0)} := \text{BPEvalH}(\text{pk}, X_x, \alpha)$. The explicit formula for X_x has depth $O(\kappa)$, and hence

$$\mathbf{GM}^{(0)} \mathbf{P}_\alpha + \mathbf{ZH}_\alpha^{(0)} = X_x(\alpha) \mathbf{G}, \quad \left\| \mathbf{H}_\alpha^{(0)} \right\|_\infty^\infty \leq (\lambda\sigma)^{O(\kappa^2)}.$$

Apply this invariant, [Equation \(54\)](#), [Equation \(55\)](#), [Lemma 3.6](#), and [Theorem 3.5](#) through the T layers, propagating every coefficient of $\mathbf{U}'$ using [Equation \(21\)](#). For every output address α_j , this yields efficiently computable matrices $\mathbf{H}_j$ and $\mathbf{K}_j$ such that

$$\mathbf{GM}^{(T)} \mathbf{P}_{\alpha_j} + \mathbf{ZH}_j + \mathbf{U}' \mathbf{K}_j = C_\lambda(x)_j \mathbf{G}, \quad (56)$$

where

$$\mathbf{U}' := [\{\mathbf{U}_{i,L,b}, \mathbf{U}_{i,R,b}\}_{i,b} \mid \{\mathbf{U}_{i,N}\}_i].$$

Set

$$\mathbf{H} := \sum_{j=0}^{n-1} \mathbf{H}_j \mathbf{e}_j, \quad \mathbf{K} := \sum_{j=0}^{n-1} \mathbf{K}_j \mathbf{e}_j.$$

Combining the equations with Equation (37) gives

$$\mathbf{GM}^{(T)}[\mathbf{P}_{\alpha_0} \mid \cdots \mid \mathbf{P}_{\alpha_{n-1}}]\mathbf{v} + \mathbf{ZH} + \mathbf{U}'\mathbf{K} = \mathbf{G}C_\lambda(x). \quad (57)$$

Each layer performs two reads on κ -bit addresses and one constant-depth gate evaluation. Starting from the initial $(\lambda\sigma)^{O(\kappa^2)}$ matrices, using $\|\mathbf{P}_\alpha\|_\infty^\infty \leq t^\kappa$, and iterating Equations (19), (25) and (27) therefore gives

$$\max\{\|\mathbf{H}\|_\infty, \|\mathbf{K}\|_\infty\} \leq (\lambda\sigma)^{O(\kappa^2 + T\kappa \log(\kappa+1))} \leq \beta_{\text{tm}}.$$

On a successful correlation, $\text{CIHash}_{\text{tm}}(\text{hk}, x) = C_\lambda(x)$. Applying $\mathbf{G}$ to Equation (50) and subtracting from Equation (57) yields

$$\mathbf{ZH} + \mathbf{U}'\mathbf{K} - \mathbf{u}_* = \mathbf{0}.$$

Here $\mathbf{H} \in \mathbb{Z}_q^{nt^2}$ and $\mathbf{K} \in \mathbb{Z}_q^{r_{\text{tm}}-1}$. Using centered integer representatives, set

$$\mathbf{w}_{\text{tm}} := \begin{bmatrix} \mathbf{H} \\ \mathbf{K} \\ -1 \end{bmatrix} \in \mathbb{Z}^{nt^2 + r_{\text{tm}}}.$$

It is a nonzero extended-DSIS solution because its last coordinate is -1 , and its norm is at most β_{tm} . Computing $\mathbf{H}$ and $\mathbf{K}$ evaluates the resulting layered circuit, whose size is polynomial in S and T , with the stated polynomial overhead. $\square$

Corollary 4.6 (Average-case depth hardness). *Let N be the key length, and suppose $N \leq 2^\kappa$. On every N -bit string K , define $g_{\lambda,T}^{\text{tm}}(K) = \text{CIHash}_{\text{tm}}(K, K)$ if K parses as a key, and define its value to be 0^n otherwise. Let*

$$L_{\text{tm}} := \{(1^\lambda, K, i) : i \in [n], g_{\lambda,T}^{\text{tm}}(K)_i = 1\}.$$

Under the decomposed-SIS parameters of Theorem 4.5, every logspace-uniform polynomial-size bounded-fan-in Boolean circuit F_λ of depth D whose layered NAND-and-copy conversion has size at most 2^κ and satisfies $3D + O(1) \leq T$ obeys

$$\Pr_{K \xleftarrow{\$} \text{KeyGen}_{\text{tm}}(1^\lambda, 1^\kappa, 1^T)} [F_\lambda(K) = g_{\lambda,T}^{\text{tm}}(K)] \leq \varepsilon(\lambda) + \delta_{\text{tail}}(\lambda)$$

which is negligible.

Proof. Convert F_λ to a layered NAND-and-copy circuit. A standard bounded-fan-in conversion has depth at most $3D + O(1)$ and polynomial size. Instantiate Theorem 4.5 with input length equal to the key length and apply Lemma 4.1. $\square$

5 The NC Hierarchy

We first bound the parallel complexity of the public hash computation for the sequential function as constructed above.

Lemma 5.1 (Parallel public evaluation). *There is a fixed integer $\rho \geq 1$ with the following property. Suppose that $\kappa = O(\log^2 N)$, that the lattice dimensions and the bit lengths of all matrix entries are polynomial in N , and that $N := \lambda + |\text{hk}| + |x|$ under the fixed encoding of $(1^\lambda, \text{hk}, x)$. Then the function*

$$(1^\lambda, \text{hk}, x) \mapsto \text{CIHash}_{\text{tm}}(\text{hk}, x)$$

is computable by logspace-uniform, polynomial-size, bounded-fan-in Boolean circuits of depth $O(T \log^\rho N)$.

Proof. Addition, multiplication, and reduction modulo q on operands of polynomial bit length, as well as products and sums of polynomial-dimensional matrices, have logspace-uniform polynomial-size Boolean circuits of depth $\log^{O(1)} N$. Kronecker products, fixed embeddings, and $\mathbf{G}^{-1}$ have the same bounds.

We apply these circuits to the explicit public algorithms above. The normalization algorithm following [Lemma 3.3](#) computes all scalar roots in parallel, evaluates each product in [Equation \(6\)](#) by a balanced tree, and then applies $\mathbf{G}^{-1}$ and the fixed arrangement specified by $\mathbf{R}_{\text{Can}}$. Hence each normalization has polylogarithmic depth. The formula for X_x in the construction has depth $O(\kappa)$, so the parallel bound in [Theorem 3.4](#) shows that $\mathbf{M}^{(0)} = \text{BPEvalM}(\text{pk}, X_x)$ has depth $\log^{O(1)} N$.

At layer i , the two calls to `BPReadM` have depth $\log(\kappa + 1) \log^{O(1)} N$ by [Lemma 3.6](#) and run in parallel. The subsequent call to `BPEvalM` evaluates the constant-depth function `Gate` and therefore also has depth $\log^{O(1)} N$ by [Theorem 3.5](#). Thus the circuit for each complete layer has polylogarithmic depth and polynomial size.

The initialization and the final computation in [Equation \(50\)](#) also have polylogarithmic depth. The only serial dependence is

$$\mathbf{M}^{(0)}, \mathbf{M}^{(1)}, \dots, \mathbf{M}^{(T)}.$$

Composing the T layer circuits gives depth $O(T \log^\rho N)$ for some fixed ρ . Its size remains polynomial because the key contains a nonempty block for every layer and hence $T \leq N$. Selecting one of the polynomially many output coordinates adds only $O(\log N)$ depth. All matrix indices and all balanced-tree connections are computable in logarithmic space, which gives the claimed uniformity. $\square$

Next, we recall the well-known collapse theorem for the NC hierarchy. This can be found, e.g., in [\[Pap94a, Chapter 15, Problem 15.5.3\]](#). We include a short proof in [Section A](#) for completeness.

Lemma 5.2 (NC collapse). *Fix $j \geq 1$. If $\text{NC}^{j+1} = \text{NC}^j$, then $\text{NC}^{j+a} = \text{NC}^j$ for every fixed $a \geq 1$. Consequently, $\text{NC} = \text{NC}^j$.*

Theorem 5.3 (Proper NC hierarchy). *Assume decomposed SIS is hard for the extension widths and solution bounds in [Theorem 4.5](#), for the parameter choices below. Then, for every $j \geq 1$,*

$$\text{NC}^j \subsetneq \text{NC}^{j+1}.$$

Proof. Fix $j \geq 1$ and set $\kappa(\lambda) := \lceil (\log \lambda)^2 \rceil$ and $T_j(\lambda) := \lceil (\log \lambda)^{j+1} \rceil$. Let L_{tm} be the language in [Corollary 4.6](#) instantiated with $T = T_j$, and let N_λ be the length of its inputs $(1^\lambda, K, i)$. Note that $\log N_\lambda = \Theta(\log \lambda)$, indeed the unary prefix gives the lower bound and the upper bound follows because T_j and κ are polylogarithmic in λ and all remaining key fields have polynomial description length. Parsing the fixed-length key encoding and returning zero on an invalid encoding has polylogarithmic depth. On valid encodings, the input length in [Lemma 5.1](#) is $O(N_\lambda)$, so that lemma gives $L_{\text{tm}} \in \text{NC}^{j+\rho+1}$, because

$$T_j \log^\rho N_\lambda = O(\log^{j+\rho+1} N_\lambda).$$

We claim that $L_{\text{tm}} \notin \text{NC}^j$. Assume not, hardwire the prefix 1^λ in its circuit and replicate the circuit for every output index $i \in [n]$. This gives a logspace-uniform polynomial-size multi-output circuit F_λ computing $g_{\lambda, T_j}^{\text{tm}}$ exactly, with depth $D_\lambda = O(\log^j N_\lambda)$. Since

$$\frac{T_j}{\log^j N_\lambda} = \Theta(\log \lambda),$$

we have $3D_\lambda + O(1) \leq T_j$ for all sufficiently large λ . Moreover,

$$\text{size}(F_\lambda) = N_\lambda^{O(1)} = \lambda^{O(1)} \leq 2^{(\log \lambda)^2} \leq 2^\kappa.$$

The key length is at most $N_\lambda \leq \lambda^c < 2^\kappa$ for all sufficiently large λ . Thus [Corollary 4.6](#) applies and gives

$$\Pr_{K \leftarrow \text{KeyGen}_{\text{tm}}(1^\lambda, 1^\kappa, 1^{T_j})} [F_\lambda(K) = g_{\lambda, T_j}^{\text{tm}}(K)] \leq \text{negl}(\lambda),$$

whereas exact computation makes this probability equal to one. This proves the claim.

Suppose now that $\text{NC}^{j+1} = \text{NC}^j$. By [Lemma 5.2](#), $\text{NC}^{j+\rho+1} = \text{NC}^j$. This contradicts $L_{\text{tm}} \in \text{NC}^{j+\rho+1}$ and $L_{\text{tm}} \notin \text{NC}^j$, therefore $\text{NC}^j \neq \text{NC}^{j+1}$. $\square$

Acknowledgements

The constructions and proofs were produced by humans. ChatGPT was used solely for writing assistance. The authors take full responsibility for all claims and proofs made in this paper.

D.A. was supported by a fellowship with the Simons Institute for the Theory of Computing. This work was supported by Input Output (iog.io) through their funding of the Edinburgh Blockchain Technology Lab. G.M. was supported by the European Research Council through an ERC Starting Grant (Grant agreement No. 101077455, ObfusQation) and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA – 390781972. L.R. was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement number 101124977 (DECRYPSIS).

References

- [ABP17] Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Depth-robust graphs and their cumulative memory complexity. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 3–32. Springer, Cham, April / May 2017. [1](#)
- [ACP⁺17] Joël Alwen, Binyi Chen, Krzysztof Pietrzak, Leonid Reyzin, and Stefano Tessaro. Scrypt is maximally memory-hard. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 33–62. Springer, Cham, April / May 2017. [1](#)
- [Ajt98] Miklós Ajtai. The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract). In *30th ACM STOC*, pages 10–19. ACM Press, May 1998. [2](#)
- [AMR25a] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Key-homomorphic computations for RAM: Fully succinct randomised encodings and more. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part III*, volume 16002 of *LNCS*, pages 236–268. Springer, Cham, August 2025. [2](#), [3](#)
- [AMR25b] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Succinct oblivious tensor evaluation and applications: Adaptively-secure laconic function evaluation and trapdoor

- hashing for all circuits. In Michal Koucký and Nikhil Bansal, editors, *57th ACM STOC*, pages 1875–1886. ACM Press, June 2025. [3](#)
- [AMR26a] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Tree encodings I: How to authenticate a truth table, 2026. [6](#), [7](#), [8](#)
- [AMR26b] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Tree encodings II: Correlation intractability for all batched relations, 2026. [4](#)
- [AS15] Joël Alwen and Vladimir Serbinenko. High parallel complexity graphs and memory-hard functions. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 595–603. ACM Press, June 2015. [1](#)
- [Ban93] Wojciech Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, 1993. [4](#)
- [BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 757–788. Springer, Cham, August 2018. [1](#)
- [BCH⁺22] Nir Bitansky, Arka Rai Choudhuri, Justin Holmgren, Chethan Kamath, Alex Lombardi, Omer Paneth, and Ron D. Rothblum. PPAD is as hard as LWE and iterated squaring. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part II*, volume 13748 of *LNCS*, pages 593–622. Springer, Cham, November 2022. [1](#)
- [BGG⁺14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Berlin, Heidelberg, May 2014. [3](#)
- [BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a Nash equilibrium. In Venkatesan Guruswami, editor, *56th FOCS*, pages 1480–1498. IEEE Computer Society Press, October 2015. [1](#)
- [CHK⁺19] Arka Rai Choudhuri, Pavel Hubáček, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. Finding a nash equilibrium is no easier than breaking Fiat-Shamir. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1103–1114. ACM Press, June 2019. [1](#)
- [CHW25] Jeffrey Champion, Yao-Ching Hsieh, and David J. Wu. Registered ABE and adaptively-secure broadcast encryption from succinct LWE. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part III*, volume 16002 of *LNCS*, pages 3–34. Springer, Cham, August 2025. [2](#)
- [CP18] Bram Cohen and Krzysztof Pietrzak. Simple proofs of sequential work. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 451–467. Springer, Cham, April / May 2018. [1](#)
- [DFKP15] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 585–605. Springer, Berlin, Heidelberg, August 2015. [1](#)

- [DLM19] Nico Döttling, Russell W. F. Lai, and Giulio Malavolta. Incremental proofs of sequential work. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 292–323. Springer, Cham, May 2019. [1](#)
- [DN93] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *CRYPTO’92*, volume 740 of *LNCS*, pages 139–147. Springer, Berlin, Heidelberg, August 1993. [1](#)
- [HLL23] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices. In *64th FOCS*, pages 415–434. IEEE Computer Society Press, November 2023. [3](#)
- [JM21] Simon Josefsson and Alexey Melnikov. The argon2 memory-hard function for password hashing and proof-of-work applications. RFC 9106, 2021. [1](#), [2](#)
- [LLL25] Hanjun Li, Huijia Lin, and George Lu. Succinct garbled circuits with low-depth garbling algorithms. *Cryptology ePrint Archive*, 2025. [2](#)
- [LT82] Thomas Lengauer and Robert E. Tarjan. Asymptotically tight bounds on time-space trade-offs in a pebble game. *SIAM Journal on Computing*, 1982. [1](#)
- [LV20] Alex Lombardi and Vinod Vaikuntanathan. Fiat-Shamir for repeated squaring with applications to PPAD-hardness and VDFs. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 632–651. Springer, Cham, August 2020. [1](#)
- [MMV11] Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. Time-lock puzzles in the random oracle model. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 39–50. Springer, Berlin, Heidelberg, August 2011. [2](#)
- [MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Berlin, Heidelberg, April 2012. [2](#)
- [Pap94a] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994. [21](#)
- [Pap94b] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *JCSS*, 1994. [1](#)
- [Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 333–342. ACM Press, May / June 2009. [2](#)
- [Per09] Colin Percival. Stronger key derivation via sequential memory-hard functions, 2009. BSDCan. [1](#), [2](#)
- [Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. [2](#)
- [RSW96] Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. 1996. [1](#), [2](#)

- [Wee25] Hoeteck Wee. Functional commitments and SNARGs for P/poly from SIS. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 617–640. Springer, Cham, August 2025. 2
- [WW25] Hoeteck Wee and David J. Wu. Unbounded distributed broadcast encryption and registered ABE from succinct LWE. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part III*, volume 16002 of *LNCS*, pages 204–235. Springer, Cham, August 2025. 2

A Proof of Lemma 5.2

Proof. We induct on a . The case $a = 1$ is the hypothesis. Suppose that $\mathbf{NC}^{j+a} = \mathbf{NC}^j$, and let $L \in \mathbf{NC}^{j+a+1}$. Let C_N be a logspace-uniform polynomial-size circuit for L of depth $O(\log^{j+a+1} N)$. Insert copy gates so that every wire joins consecutive layers, and pad every layer to the same width $W = N^{O(1)}$, with $W \geq N$. These changes preserve polynomial size and logspace uniformity.

Partition C_N into $s = O(\log N)$ slabs, each of depth $O(\log^{j+a} N)$. For a slab index u and a boundary state $z \in \{0, 1\}^W$, let

$$\Phi_N(u, z) \in \{0, 1\}^W$$

be the state at the next boundary. The language

$$\{(u, z, b) : \Phi_N(u, z)_b = 1\}$$

belongs to $\mathbf{NC}^{j+a}$: Evaluate all slab circuits on z in parallel and select the slab and output coordinate specified by u and b . The selection adds depth

$$O(\log(sW)) = O(\log N),$$

so the total depth remains $O(\log^{j+a} N)$, and the total size is polynomial. Including u as an input also makes this family logspace-uniform.

By the inductive hypothesis, this language belongs to $\mathbf{NC}^j$. Hardwiring a slab index and replicating its circuit for all W output coordinates gives a polynomial-size, depth- $O(\log^j W) = O(\log^j N)$ multi-output circuit for Φ_N . Hardwire the successive slab indices and compose $s = O(\log N)$ copies. The resulting circuit computes C_N and has depth

$$O(\log N \cdot \log^j N) = O(\log^{j+1} N).$$

Thus $L \in \mathbf{NC}^{j+1} = \mathbf{NC}^j$, completing the induction. Replication, hardwiring, and composition preserve logspace uniformity. $\square$