Homomorphism in Expectation: Leveled Homomorphic Encryption with Range-Relative Correctness

Zeyu Liu[3] zeyu.liu@yale.edu
Xianhui Lu[1,2] luxianhui@iie.ac.cn
Ruida Wang[1,2*] wangruida@iie.ac.cn
Xinxuan Zhang[1,2] zhangxinxuan@iie.ac.cn

[1]State Key Laboratory of Cyberspace Security Defense,
Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]Yale University, New Haven, United States

Abstract

Lattices have dominated the development of homomorphic encryption for deep computation. We introduce *Homomorphism in Expectation (HIE)*, a new construction showing that encrypted addition suffices for both addition and multiplication in deep real-valued computation.

Given any additively homomorphic encryption scheme over $\mathbb{Z}_2$, we construct range-relatively correct leveled homomorphic encryption for arithmetic circuits C of *arbitrary* polynomial depth. For depth bound d and any chosen public input bound $r > 0$, our scheme supports inputs in $[-r, r]$. Outputs lie in $[-\eta_C, \eta_C]$, where $\eta_C = C(r, \ldots, r)$ is the circuit's maximum output range magnitude over this input range. Decryption returns an answer with error at most $\varepsilon \cdot \eta_C$ except with probability δ. Each input requires $\ell_{\mathsf{in}} = O(d + \log(1/\varepsilon))$ base ciphertexts, and each output requires $\ell_{\mathsf{out}} = O(\varepsilon^{-2} \log(1/\delta))$ base ciphertexts. For n inputs, online evaluation uses only $\sim\ell_{\mathsf{out}} \cdot n \cdot \ell_{\mathsf{in}}$ ciphertext additions and $\sim\ell_{\mathsf{out}} \cdot |C| \cdot \ell_{\mathsf{in}}$ memory accesses.

Our construction begins by encoding real-valued inputs as encrypted bitstreams, allowing multiplication through encrypted bit addition. The direct approach incurs costs exponential in circuit depth, limiting efficient evaluation to logarithmic depth and requiring non-compact input ciphertexts. We overcome these limitations with new encoding and circuit evaluation techniques, which eliminate the exponential overhead, enabling compact homomorphic arithmetic at arbitrary polynomial depth from additive encryption alone.

*Ruida Wang initiated the research topic and serves as the corresponding author (wangruida@iie.ac.cn).

Contents

1 Introduction **3**
- 1.1 Our Contributions . . . 3
- 1.2 Related Work . . . 4

2 Technical Overview **7**
- 2.1 Bipolar Bernoulli Encoding . . . 7
- 2.2 Scaling for Input and Addition . . . 9
- 2.3 Compact Input from Binary Probability Encoding . . . 9
- 2.4 Arbitrary Depth from a New Evaluation Procedure . . . 10
- 2.5 Additional Discussion . . . 12

3 Preliminaries **12**
- 3.1 Notation . . . 12
- 3.2 Arithmetic Circuits . . . 13
- 3.3 Additively Homomorphic Encryption . . . 13
- 3.4 Probability Tools . . . 14

4 Definition **15**

5 Main Construction **17**
- 5.1 The Ideal Construction . . . 17
- 5.2 The Finite-Precision Instantiation of the Ideal Construction . . . 22
- 5.3 Additional Discussion . . . 25
  - 5.3.1 Tighter Bound on η_C . . . 26
  - 5.3.2 Extending HIE to $\mathbb{Z}_p$ . . . 27
  - 5.3.3 Moving Public Evaluation Offline . . . 28

6 Concluding Remarks **28**

Acknowledgment **29**

References **30**

A Finite Binomial Sampling **34**

B Toy Example **35**

1 Introduction

Homomorphic encryption (HE) enables an untrusted server to compute on encrypted data without learning the inputs. In leveled HE, the client fixes a depth bound at setup and encrypts its inputs once. The server can then evaluate any supported circuit within that bound and return a compact encrypted answer, which the client decrypts without repeating the computation.

Achieving leveled HE, however, requires navigating a tension between cryptographic privacy and algebraic expressiveness. Historically, nearly all designs have addressed this by seeking hardness assumptions with richer algebraic structures, most notably lattices, which blend ring arithmetic with noise approximation. Since Gentry's breakthrough [Gen09], lattices have dominated the development of HE for deep computation. Developing a fundamentally different foundation for deep homomorphic computation remains a compelling objective. Obfuscation offers a non-lattice route [CLTV15, JLS22], but demands powerful cryptographic machinery, strong assumptions, and prohibitive concrete costs. This motivates our central question:

Can compact leveled HE for polynomial-depth circuits be built from lightweight cryptographic primitives?

We take a step toward this goal by constructing compact leveled HE for polynomial-depth arithmetic circuits from *additively homomorphic encryption*, under the relaxed notion of *range-relative correctness*.

1.1 Our Contributions

Main theorem (informal). *Any additively homomorphic encryption Π_A over $\mathbb{Z}_2$ yields range-relatively correct leveled homomorphic encryption Π for arithmetic circuits of arbitrary polynomial depth d. For any chosen public input bound $r > 0$, the input range is $[-r, r]$; the correctness is measured relative to the circuit's maximum output magnitude $\eta_C = C(r, \ldots, r)$.*

We introduce *Homomorphism in Expectation (HIE)*, a new construction showing that *addition of encrypted bits suffices for deep real-valued addition and multiplication* with relaxed correctness. This result also extends to $\mathbb{Z}_p$ with a moderately sized modulus p; see Section 5.3.2 for details.[1]

The HIE construction combines (1) a new formalized range-relative correctness notion over a predetermined input range, (2) compact input and output ciphertexts, and (3) simple and efficient encrypted computation using only the base scheme's additive operation.

(1) Range-relative correctness and predetermined input range. Let $0 < \varepsilon \leq 1$ be the target *relative* error, $0 < \delta < 1/2$ be the allowed failure probability, and λ be the security parameter. For a supported circuit C and input vector $\vec{x}$ and let η_C denotes the maximum magnitude of output range over the entire input range[2], decryption returns an estimate res satisfying

$$\Pr[|\mathsf{res} - C(\vec{x})| > \varepsilon \cdot \eta_C] \leq \delta + \mathsf{negl}(\lambda).$$

Each input lies in the chosen range $[-r, r]$ where $r > 0 \in \mathbb{Q}$, and each output lies in $[-\eta_C, \eta_C]$. The error scales with the circuit's output range: it is at most an ε fraction of η_C. Smaller output ranges therefore give smaller absolute error proportionally.

[1] Alternatively, suitable multiplicatively homomorphic encryption also suffices. See Remark 5.4 for details.

[2] $\eta_C = C(r, r, \ldots, r)$ since the circuit contains only additions and multiplication. To represent a subtraction in the circuit, one could use a public negative input. See Section 5.3.1 for a detailed discussion.

Definition 4.1 gives the formal definition of our *range-relatively correct* HE scheme, and Remark 4.2 further compares this new correctness definition with more standard correctness definitions of existing HE schemes.

(2) Compact inputs and outputs. Each input ciphertext of Π consists of $\ell_{\mathsf{in}} + 1$ ciphertexts of Π_A: ℓ_{in} probability digits and one complementary bit, where

$$\ell_{\mathsf{in}} = d + \lceil \log_2(64/\varepsilon) \rceil + 1 = O(d + \log(\varepsilon^{-1})).$$

Each output ciphertext contains ℓ_{out} ciphertexts of Π_A, where

$$\ell_{\mathsf{out}} = \left\lceil 16\varepsilon^{-2} \lceil \log_2(8/\delta) \rceil \right\rceil = O\Big(\varepsilon^{-2} \log(1/\delta)\Big).$$

Input length grows only with the declared depth and logarithmically with inverse accuracy; output length is independent of the circuit.[3]

(3) Simple and efficient computation. Let n be the number of real-valued inputs and $|C|$ the number of arithmetic gates. Evaluation requires roughly

$$\begin{aligned} &\ell_{\mathsf{out}} \cdot \ n \ \cdot \ell_{\mathsf{in}} \text{ ciphertext additions using } \Pi_A \\ &+ \ell_{\mathsf{out}} \cdot |C| \cdot \ell_{\mathsf{in}} \text{ memory accesses (or binomial draws).}[4] \end{aligned}$$

Evaluation consists of three steps:

- **One public arithmetic pass.** Evaluate C once on a public reference input, using $|C|$ bounded-precision arithmetic operations in plaintext.
- **Addition gates:** Let $A \le |C|$ be the number of addition gates. Across the ℓ_{out} output samples, these gates require at most $\ell_{\mathsf{out}} A$ memory reads (or binomial draws).
- **Multiplication gates:** The $|C| - A$ multiplication gates require at most $\ell_{\mathsf{out}} \cdot n \cdot \ell_{\mathsf{in}}$ further memory reads (or binomial draws) and at most $\ell_{\mathsf{out}} \cdot n \cdot (\ell_{\mathsf{in}} + 1)$ ciphertext additions.

In total, evaluation uses $|C|$ plaintext arithmetic operations, at most $\ell_{\mathsf{out}} \cdot (A + n \cdot \ell_{\mathsf{in}})$ memory reads or binomial draws, and at most $\ell_{\mathsf{out}} \cdot n \cdot (\ell_{\mathsf{in}} + 1)$ ciphertext additions. Furthermore, unlike existing lattice-based solutions for leveled HE, HIE does not require any evaluation key (unless required by the underlying Π_A), which is also a major communication bottleneck of existing leveled HE schemes. A preliminary implementation for private MNIST inference shows promising concrete efficiency; see Section 6 for details.

1.2 Related Work

This section surveys related work and compares them in Table 1.

Additive encryption and limited multiplication. Additive encryption is available under diverse assumptions: quadratic residuosity for Goldwasser–Micali [GM84] and Joye–Libert schemes [JL13, BHJL17]; higher residuosity for Benaloh [Ben94, FLA11] and Naccache–Stern [NS98];

[3] For our finite-precision construction, the public decoding scale uses $O(\lambda + d + \log(1/\varepsilon) + \log(L_r + 1))$ bits, where L_r is the bit length of the public input bound r. However: (1) ℓ_{out} still remains independent of d and (2) η_C is independent of the input, thereby precomputable.

[4] All public sampling choices can be generated and stored in advance, independently of the real-valued inputs, replacing online sampling with memory reads. See Section 5.3.3 for details.

Type	Assumption	Plaintext domain	Computation	Restriction	Evaluation key size
AHE	DDH, QR, DCR, Lattice, etc.[1]	Additive group	Linear functions	No encrypted multiplication	—
SHE/LHE	Subgroup decision[2]	Residue ring	Quadratic polynomials	$D \le 1$	—
	Any AHE[3]	Ring	Quadratic polynomials	$D \le 1$ Output: $\Omega(\lambda t)$	—
	Sparse LPN + suitable AHE[4]	Residue ring	Polynomials with M monomials	$D \le \log_2\left(\frac{\log \lambda}{\log\log \lambda}\right)$ $M \le \mathsf{poly}(\lambda)$	$\mathsf{poly}(\lambda, M(\log \lambda)^{2^D})$
	DCR[5]	Boolean	Branching programs	$\ell \le \mathsf{poly}(\lambda)$ Output: $\Omega(\lambda \ell)$	—
	Subexponential DDH[6]	Boolean	Threshold circuits	$D = O(1)$; high decryption error	$\mathsf{poly}(\lambda, w)$
	Lattice[7]	Ring	Arithmetic circuits	$D \le \mathsf{poly}(\lambda)$	$\mathsf{poly}(\lambda)$
	Any AHE[8]	Real-valued interval	Range-relative arithmetic circuits	$D \le \mathsf{poly}(\lambda)$ $\varepsilon^{-1}, \log(1/\delta) \le \mathsf{poly}(\lambda)$	—
FHE	Lattice + circular security[9]	Ring	Arbitrary circuits	—	$\mathsf{poly}(\lambda)$
	Probabilistic iO + suitable PKE[10]	Boolean	Arbitrary circuits	—	$\mathsf{poly}(\lambda)$

Table 1: Representative HE constructions. Evaluation-key sizes exclude the public key; an em dash (—) means none. D is binary multiplicative depth for the quadratic and sparse-LPN rows, and total depth otherwise; ℓ is the branching program length, M the monomial count, w the threshold-circuit width, and t is the number of quadratic products in a quadratic polynomial. AHE means additive HE; SHE means somewhat HE; LHE means leveled HE; FHE means fully HE.

[1] AHE is available from QR [GM84, JL13, BHJL17], higher residuosity [Ben94, FLA11, NS98], the p-subgroup assumption [OU98], DCR [Pai99, DJ01], DDH [BCP03, CL15], class-group subgroup membership [CLT22], and (R)LWE [Reg09, LPR13].

[2] BGN supports one multiplication layer under subgroup decision in composite-order bilinear groups, with efficient decryption for bounded outputs [BGN05].

[3] Catalano and Fiore use linearly homomorphic encryption over a publicly known finite commutative ring with identity and efficient uniform sampling to support one multiplication layer [CF15]. Single-server compactness holds for a constant number of products of linear forms; two noncommunicating, noncolluding servers permit arbitrary quadratic polynomials.

[4] The displayed sufficient depth bound follows from degree at most 2^D. It applies to the secret-key construction of [CGHTV25]; its public-key extension supports constant degree. The AHE must support the required plaintext linear operations; DDH and DCR provide instantiations. The evaluation-key bound is a coarse upper bound for logarithmic sparsity; the key is reusable.

[5] The Ishai–Paskin construction uses Damgård–Jurik encryption; output length and decryption work depend on program length, rather than the total number of nodes [IP07].

[6] Jain and Jin allow very high decryption failure after evaluation. For constant depth, the evaluation key has size $\mathsf{poly}(\lambda, w)$; exponential DDH permits threshold depth $O(\log \lambda)$ [JJ25].

[7] Lattice-based HE includes (R)LWE [BGV14, Bra12, GSW13, CKKS17], approximate-GCD schemes [vDGHV10] connected to LWE [CS15], and lattice isomorphism [BMM26, BLSD26]. The evaluation-key size illustrates BGV; leveled GSW needs none.

[8] **This work.** Compatible bit- or integer-additive schemes give instantiations under the assumptions in note 1. Error is bounded by $\varepsilon\eta_C$ for error δ and output range bound η_C, except with probability $\delta + \mathrm{negl}(\lambda)$; η_C is the public output bound. We can use any AHE over $\mathbb{Z}_2$ or $\mathbb{Z}_p$ for sufficiently large p (see Section 5.3.2), or multiplivative HE (see Remark 5.4).

[9] Representative lattice FHE constructions additionally assume circular security [Gen09, BV14, BGV14, GSW13, BMM26, DM15, CGGI20]. Hsiao, Lee, Reagen, and Alaghi evaluate stochastic bitstreams using TFHE [HLRA22].

[10] Probabilistic obfuscation and suitable encryption yield FHE without circular security [CLTV15]. Non-lattice iO constructions use subexponentially secure large-field LPN, the decisional linear assumption, and polynomial-stretch PRGs in NC^0 [JLS22], or replace the PRGs with binary sparse LPN [RVV24].

the p-subgroup assumption for Okamoto–Uchiyama [OU98]; and decisional composite residuosity for Paillier [Pai99] and Damgård–Jurik [DJ01]. DDH-based constructions include Bresson–Catalano–Pointcheval [BCP03] in RSA groups and Castagnos–Laguillaumie [CL15] in class groups. Castagnos–Laguillaumie–Tucker [CLT22] additionally provide power-of-two plaintext spaces under a class-group subgroup-membership assumption. LWE [Reg09] and Ring-LWE [LPR13] encryption also support additions. Beyond addition, Boneh–Goh–Nissim [BGN05] support one multiplication layer under the subgroup-decision assumption in composite-order bilinear groups, with efficient decryption for bounded plaintexts.

With suitable plaintext spaces and parameters, all these families provide instantiations of HIE through their bit or integer-additive interface, including the lattice-based ones, which only provide weakly additive homomorphism (see Remark 3.2 for details).

Transformations of linear homomorphism. Catalano and Fiore [CF15] add one multiplication level to linearly homomorphic encryption over publicly known rings admitting uniform sampling. Their outputs are compact for a constant number of summed quadratic terms with one server, or arbitrary quadratic polynomials with two noncommunicating, noncolluding servers.

Corrigan-Gibbs, Henzinger, Kalai, and Vaikuntanathan [CGHTV25] combine suitable linearly homomorphic encryption with sparse LPN to support $O(\log \lambda / \log \log \lambda)$ multiplications followed by polynomially many additions, where λ is the security parameter. Ciphertexts remain compact, while a reusable evaluation key grows with the complexity of the supported computation.

Encrypted branching-program evaluation. Ishai and Paskin [IP07] evaluate branching programs on encrypted inputs using nested encryption, with an efficient Damgård–Jurik instantiation. Client costs avoid dependence on the full program size, but the returned ciphertext length and decryption work still depend on the program length.

Lattice-based leveled and fully homomorphic encryption. Gentry [Gen09] established fully homomorphic encryption through bootstrapping. Subsequent lattice-based constructions include BGV [BGV14], BFV [Bra12, FV12], GSW [GSW13], FHEW [DM15], TFHE [CGGI20], and many follow-up works (e.g., [LMP22, LMK+23, LW23, WWL+24, LW24, WHS+25, JLLS26]). Other constructions use approximate GCD [vDGHV10], connected to LWE by Cheon and Stehlé [CS15], or lattice isomorphism [BMM26, BLSD26].

Cheon, Kim, Kim, and Song [CKKS17] introduced CKKS for approximate arithmetic in lattice ciphertexts, using rescaling to manage precision. Our construction instead represents numbers through encrypted distributions, implements multiplication through encrypted XOR, and recovers the answer by averaging $\ell_{\mathsf{out}} = O(\varepsilon^{-2} \log(1/\delta))$ output stream bits.

Obfuscation-based alternatives. Canetti, Lin, Tessaro, and Vaikuntanathan [CLTV15] construct FHE from probabilistic obfuscation and suitable encryption without a circular-security assumption. Non-lattice obfuscation constructions by Jain, Lin, and Sahai [JLS22] use subexponentially secure large-field LPN, the decisional linear assumption, and polynomial-stretch PRGs in NC^0; Ragavan, Vafa, and Vaikuntanathan [RVV24] replace the last assumption with binary sparse-LPN. These approaches rely on heavy general-purpose obfuscation machinery, whereas ours uses additive encryption directly for approximate arithmetic on a bounded domain.

Relaxing correctness to obtain alternative foundations. Jain and Jin [JJ25] construct sometimes-decryptable HE from DDH, supporting constant-depth threshold circuits under subexponential hardness and logarithmic-depth threshold circuits under exponential hardness. Their relaxation permits high decryption failure probability; ours permits numerical error while retaining correctness within a prescribed tolerance with high probability.

Homomorphic encryption for stochastic computing. Originated in the 1960s [Gai67], stochastic computing (SC) represents continuous values via random bitstreams to enable low-cost

Symbol	Meaning
Π_A	Underlying additively homomorphic encryption scheme
λ	Security parameter
d	Circuit depth bound
$n, \lvert C\rvert, A$	Number of inputs, arithmetic gates, and addition gates
ε, δ	Relative error tolerance and failure probability
$r > 0$	Chosen public input bound with bit length L_r; $x_i \in [-r, r]$
ℓ	Number of samples in a stream
ℓ_{in}	Number of encrypted probability digits per input
ℓ_{out}	Number of encrypted bipolar samples per output
η_v, η_C	Node v weights and root weight (maximum output magnitudes)
$k_i, k_{i,j}$	Sample requests for input i and selections of its digit j
$x, u = r^{-1} \cdot x$	Input and normalized input
$q = \frac{1-u}{2}$	Probability of sampling -1 for a Bernouli variable in a stream

Table 2: Variables used in HIE.

arithmetic; after decades of dormancy, it has experienced a notable resurgence since the 2010s driven by AI acceleration, in-memory computing, and memristor architectures. Recent works such as [KIU+23] and [HLRA22] bridge SC with homomorphic encryption, yet from a different perspective than ours. Specifically, Hsiao, Lee, Reagen, and Alaghi [HLRA22] investigate how to evaluate conventional stochastic operations using existing FHE schemes; Koseki, Ito, Ueno, Tibouchi, and Homma [KIU+23] focus on using HE to realize SC computation, instantiating an HE-based secure computation scheme, termed HE for stochastic computing (HESC). In contrast, despite sharing certain similarities, our general goal is to lift AHE directly into a compact HE scheme for deep arithmetic circuits with SC as a tool.

Beyond this divergence in research goals, our encoding and evaluation mechanisms depart from standard SC. While Section 2.1 begins with the classic bipolar bitstream intuition used in [KIU+23], as discussed in Section 2.1, directly composing explicit bitstreams in general circuits suffers from scaling mismatch, sample dependence across reconvergent paths, and an $\Omega(2^d)$ blowup in stream length and computation, making it far from being sufficient for the leveled HE functionality we aim for. Therefore, additional techniques and formalized proofs would be necessary. Furthermore, [KIU+23] does not establish correctness or compactness guarantees for general arithmetic circuits, which is why we omit it from Table 1. Our construction addresses these issues and provides explicit accuracy and efficiency bounds for the supported circuit families.

2 Technical Overview

2.1 Bipolar Bernoulli Encoding

We start by recalling how stochastic encoding (as shown in [KIU+23]) can be used to separate additions and multiplications using special encoding and by relaxing correctness to *range-relative correctness*, as detailed below. See Table 2 for a summary of the variables we use.

Setup. The circuits have input variables as leaves and binary addition and multiplication gates. For now, assume both the inputs and outputs lie in $[-1, 1]$. We will handle more general inputs $[-r, r]$ later.

Fixing the depth and accuracy parameters and encrypting each input once, for a supported circuit C and input vector $\vec{x}$, decryption returns an estimate res satisfying

$$\Pr[|\mathsf{res} - C(\vec{x})| > \varepsilon \cdot \eta_C] \leq \delta + \mathsf{negl}(\lambda).$$

We start by explaining how we encode a value in $[-1, 1]$, which is central to how we perform the computation later.

Encoding a number as an expectation. Inspired by stochastic encoding [Gai67], same as the encoding in [KIU+23], we represent a number as a stream $S := (S_1, \ldots, S_\ell)$ for some length ℓ of independent samples from $\{-1, 1\}$ and encrypt the corresponding bits. The point of this stream encoding is to "separate" multiplication from addition, as detailed below.

For each $i \in [\ell]$, let $S_i \in \{-1, 1\}$ be a bipolar Bernoulli variable satisfying $\mathbb{E}[S_i] = x$, or equivalently,

$$\Pr[S_i = 1] = \frac{1 + x}{2}.$$

We encode each sample as a bit b_i with $S_i = (-1)^{b_i}$: $+1$ corresponds to $b_i = 0$, and -1 corresponds to $b_i = 1$. We then encrypt b_i under Π_A.

Decoding averages ℓ independent decrypted samples. For any $t > 0$, Hoeffding's inequality gives

$$\Pr\left[\left|\frac{1}{\ell}\sum_{j=1}^{\ell} S_j - x\right| > t\right] \leq 2\exp(-\ell t^2/2).$$

Thus $\ell = O(\varepsilon^{-2}\log(1/\delta))$ samples suffice to achieve absolute error at most ε for the encoded mean, except with probability δ (see the main theorem for the explicit constants).

Multiplication via XOR. One useful property of this encoding is that multiplication immediately becomes XOR. For independent samples $S_i = (-1)^{b_i}$ and $S'_i = (-1)^{b'_i}$ representing x and y, respectively, we have

$$S_i S'_i = (-1)^{b_i \oplus b'_i},$$

and therefore

$$\mathbb{E}[S_i S'_i] = xy.$$

Importantly, S_i and S'_i must be independent. XORing a stream with itself gives an all-zero bit stream, which represents the constant sign $+1$, rather than x^2. This creates an additional issue that we discuss below.

"Lossy" addition via averaging. Computing $x + y$ is trickier. However, we can form a new stream by randomly selecting one of the two samples at each position: for every $i \in [\ell]$, set $S''_i = S_i$ or $S''_i = S'_i$, each with probability $1/2$. The resulting mean is $(x + y)/2$, giving us a "lossy" addition.

Issues left. This direct encoding procedure is not sufficient. There are three issues:

- We do not handle the general input range $[-r, r]$, and addition incurs a factor-$\frac{1}{2}$ loss.
- Independent sampling can require 2^d fresh samples of the same input for a single output sample, since an input may occur 2^d times in the expanded computation.
- Naively evaluating that expanded computation can also require 2^d operations per output sample.

Unfortunately, these issues are not formalized nor addressed in [KIU+23], thereby making it unclear whether such an encoding scheme can be used for deep real-valued computation. Next, we address these three issues one by one.

2.2 Scaling for Input and Addition

Scale the input. To handle general inputs in $[-r, r]$, instead of directly encoding and encrypting x, we first scale it to $r^{-1}x$, where $r > 0$ is a chosen public input bound and $|x| \le r$. Each addition between two streams at a common scale halves that scale. We can also keep track of the factor $1/2$ scale from additions, and resolve it during decryption. This is handled more directly when we handle the overal evlaution process in Section 2.4.

Uneven scaling. Multiplication, however, multiplies the scales: two streams at scale r^{-1} produce a stream at scale r^{-2}. Addition requires its two input streams to have the same scale. One way to align them is to mix the stream at the larger scale with independent fair bipolar samples, whose mean is zero, until its scale matches the other stream's.

At the output, we divide the average of the decrypted samples by the remaining scale. Later, we introduce an alternative evaluator that uses a tighter public normalization and makes evaluation more efficient.

Scaling resolves the first issue listed at the end of Section 2.1. This already gives range-relatively correct leveled HE for logarithmic-depth arithmetic circuits (i.e., NC^1).

However, the remaining two issues make this direct approach exponentially expensive in the depth, preventing efficient evaluation at arbitrary polynomial depth.

Furthermore, the input streams are also not compact: their lengths can grow exponentially with the depth. This makes outsourcing less attractive, since the client can spend more work preparing and encrypting the inputs than evaluating the circuit in plaintext. To obtain compact inputs and support polynomial-depth circuits efficiently, we therefore need to address the remaining two issues.

2.3 Compact Input from Binary Probability Encoding

We start with the second issue: reducing the input size to be almost independent of the circuit size.

Encrypting the expectation. Instead of encrypting many samples, the client encrypts the probability distribution that describes the "expected value": recall that for the bitstream, given input $x \in [-r, r]$, let $u = r^{-1}x \in [-1, 1]$ be the normalized input; we want $\mathbb{E}[S_i] = u$, or equivalently,

$$q := \Pr[S_i = -1] = \frac{1 - u}{2}.$$

The encryption encrypts this value q in bits in its binary representation.

We store only ℓ_{in} binary digits of q, where ℓ_{in} is the number of probability digits and also serves as the precision parameter: more digits give a more accurate representation, but require more input ciphertexts.

Choose the probability digits $b_1, \ldots, b_{\ell_{\mathsf{in}}}$ and the complementary bit σ as follows:

- If $q < 1$, set $\sigma = 0$ and use the first ℓ_{in} fractional binary digits of q, so
$$\lfloor 2^{\ell_{\mathsf{in}}} q \rfloor = \sum_{j=1}^{\ell_{\mathsf{in}}} b_j 2^{\ell_{\mathsf{in}} - j}.$$
- If $q = 1$ (or equivalently, $u = -1$), set $\sigma = 1$ and every digit $b_j = 1$.

The client encrypts all these bits. The represented probability is

$$\tilde{q} = \sum_{j=1}^{\ell_{\mathsf{in}}} b_j 2^{-j} + \sigma 2^{-\ell_{\mathsf{in}}}.$$

To guarantee sufficient output accuracy, it suffices to choose $\ell_{\mathsf{in}} = O(d + \log(1/\varepsilon))$ where d is the circuit depth, and ε is the desired output accuracy. Correctness still holds because, intuitively, truncation changes each input (normalized by r^{-1}) by less than $2^{1-\ell_{\mathsf{in}}}$, while each circuit level increases the error in (normalized) intermediate values by at most a factor of two. This choice therefore bounds the contribution from input rounding by $O(\varepsilon)$ in the normalized output, or $O(\varepsilon \cdot \eta_C)$ after rescaling (discussed below).

Resampling from one record. The server can now generate a sample by randomly selecting one of the encrypted digits. Specifically, it selects digit j with public probability 2^{-j}.

With the remaining probability $2^{-\ell_{\mathsf{in}}}$, it selects the encrypted complementary bit σ. The selected bit B is therefore one with probability exactly

$$\sum_{j=1}^{\ell_{\mathsf{in}}} b_j 2^{-j} + \sigma 2^{-\ell_{\mathsf{in}}} = \tilde{q}.$$

The selected bit represents the bipolar sample $S = 1 - 2B = (-1)^B$, with mean

$$\tilde{u} = 1 - 2\tilde{q},$$

where $\tilde{u}$ is the rounded version of $u = r^{-1}x$.

Each new sample uses an independent random selection. The server can therefore reuse the same encrypted record whenever another sample is needed.

This resolves the second issue: the client encrypts one short probability record, and the server generates independent samples from it as needed.

2.4 Arbitrary Depth from a New Evaluation Procedure

The last issue seems harder to overcome: each use of an input requires a fresh sample. In a circuit of depth d, an input may be used $\Omega(2^d)$ times, requiring $\Omega(2^d)$ fresh samples. This computational cost seems inherent.

However, we observe that we can instead take a "top-down" view of the circuit, collect all the sample requests, and sample them together rather than individually. We first describe the high-level construction below. See Section B for a simple toy example.

A public normalization for every gate. We start by assigning a "normalization weight" to each gate by evaluating C once with every input set to the fixed public reference value r. For the subcircuit C_v ending at node v, record

$$\eta_v = C_v(r, \ldots, r), \qquad \eta_C = \eta_{\text{root}}.$$

Input weights are r; addition adds weights, and multiplication multiplies them.

Since the circuit uses only addition and multiplication of input variables, these weights bound the magnitudes of the corresponding values. In particular,

$$\eta_C = \max_{\vec{x} \in [-r,r]^n} |C(\vec{x})| = C(r, \ldots, r).$$

We use these weights to normalize every intermediate value and undo the final normalization when decoding. Note that this step uses a fixed public dummy evaluation and requires no information about the encrypted inputs.

One conceptual output sample. With these public weights, the evaluator traverses the circuit from the top (output) layer to the bottom (input) layer as follows.

It initializes a to-do list containing the root (the circuit itself) and applies the following rules.

- At an addition, select the left child with public probability $p_v = \eta_L/(\eta_L + \eta_R)$, and the right child with probability $1 - p_v$. Replace the current item with the selected child.
- At a multiplication, replace the current item with both children. If both edges point to the same child, include it twice.
- Visit the next non-input item on the to-do list. Apply the corresponding addition or multiplication rule, and continue until the list contains only inputs.
- For each input occurrence in the list, generate a fresh encrypted bipolar sample from that input's reusable probability record. Repeated occurrences of the same input use independent selections. Multiply all the resulting samples by homomorphically XORing these bits.

Repeat this process independently ℓ_{out} times. To decrypt, average the resulting stream bits $S_1, \ldots, S_{\ell_{\mathsf{out}}}$ and multiply by η_C:

$$\mathsf{res} = \frac{\eta_C}{\ell_{\mathsf{out}}} \sum_{j=1}^{\ell_{\mathsf{out}}} S_j.$$

Since decoding multiplies the average by η_C, it also multiplies the sampling error by η_C.

Thus, the number of ciphertexts per output is

$$\ell_{\mathsf{out}} = \left\lceil 16\varepsilon^{-2} \left\lceil \log_2(8/\delta) \right\rceil \right\rceil = O\Big(\varepsilon^{-2} \log(1/\delta)\Big).$$

The same output size given ε, δ gives error at most $\varepsilon \cdot \eta_C$ for every supported circuit, which means smaller output ranges give proportionally smaller absolute error.

Why the result is correct. Intuitively, this evaluation procedure simply reverses the direction of the usual bottom-up circuit evaluation. Addition is implemented by random selection using the public weights, which accounts for the different normalizations of its two children. Multiplication remains a product of independent samples. More precisely, in the ideal sampling procedure, a sample generated from node v has mean

$$\frac{C_v(\tilde{\vec{x}})}{\eta_v},$$

where $\tilde{\vec{x}}$ is the vector of inputs represented by the stored (rounded) probability records. The weighted choice at an addition preserves this relation, as does multiplying independent samples at a multiplication. At the root, each sample therefore has mean $C(\tilde{\vec{x}})/\eta_C$. Multiplying the average by η_C restores the desired value $C(\tilde{\vec{x}})$. The choices of ℓ_{in} and ℓ_{out} bound the input-rounding and sampling errors by $\varepsilon \cdot \eta_C$ in total; see Section 5 for the proof.

Sampling together. Changing the direction of evaluation alone may still require 2^d computation time: the to-do list may contain 2^d items!

However, repeated occurrences of the same gate refer to the same node (including an input node) in the public circuit. We therefore keep one counter per node. Whenever another sample is requested from that gate, we increase its counter instead of adding another copy to the list. To start, we initialize the root's counter to one and all other counters to zero. We process nodes from the output toward the inputs, with every parent processed before its children. By the time a node is processed, all requests from its parents have been added to its counter.

At a multiplication with counter c, add c to each child's counter, including twice when both edges point to the same child. At an addition node v with counter c, draw $J \sim \mathrm{Bin}(c, p_v)$, meaning the number of its c requests routed to the left child. Add J to the left child's counter and $c - J$ to the right child's counter.

Thus, we process each stored gate at most once. The binomial draw represents all its independent choices together; we do not generate one sample and reuse it. The resulting input counter k_i tells us how many samples are needed from x_i, without enumerating those requests. Each counter can reach 2^d, but takes only $d+1$ bits to store.

Recall that multiplications are simply XORs. Rather than generating the samples one by one, we record how many samples select each bit of the input's probability record. For example, suppose input x_i has probability digits $b_{i,j}$ and must be sampled k_i times. Let $k_{i,j}$ be the number of times digit $b_{i,j}$ is selected. Then

$$\sum_{j=1}^{\ell_{\mathsf{in}}} k_{i,j} + k_{i,0} = k_i,$$

where $k_{i,0}$ counts selections of the encrypted complementary bit σ_i, which is chosen with probability $2^{-\ell_{\mathsf{in}}}$.

These counts can also be sampled together. First draw $k_{i,1} \sim \mathrm{Bin}(k_i, 1/2)$, subtract that count, and repeat with the remaining requests for the next digit. Each step uses probability $1/2$ because, among selections that have not chosen an earlier digit, half choose the next digit. After ℓ_{in} steps, the remaining requests select the complementary bit, so their number is $k_{i,0}$.

Since these bits are eventually XORed together, digit $b_{i,j}$ contributes zero when $k_{i,j}$ is even, and contributes $b_{i,j}$ when $k_{i,j}$ is odd. We also include the encrypted complementary bit σ_i exactly when its own selection count $k_{i,0}$ is odd. This replaces $k_i = O(2^d)$ individual samples with ℓ_{in} binomial draws and at most $\ell_{\mathsf{in}}+1$ ciphertext operands per input.[5]

Resulting scheme. One public pass uses $|C|$ arithmetic operations. Let A be the number of addition gates and n the number of real-valued inputs. Across all output samples, the count procedures use at most $\ell_{\mathsf{out}}(A + n\ell_{\mathsf{in}})$ binomial calls, and the final encrypted combinations use at most $\ell_{\mathsf{out}} n(\ell_{\mathsf{in}}+1)$ ciphertext operands. The input and output sizes are respectively $\ell_{\mathsf{in}}+1$ and ℓ_{out} base ciphertexts per scalar, as in the introduction.

Putting everything together, we obtain the leveled homomorphic encryption scheme in our main theorem, with range-relative correctness and polynomial-time evaluation for circuits of arbitrary polynomial depth.

2.5 Additional Discussion

Public inputs can help tighten η_C and improve accuracy: their known magnitudes replace the worst-case bound r, while negative values may allow further reductions through cancellation. Our HIE also extends to additive encryption over $\mathbb{Z}_p$, provided p is large enough to prevent wraparound in the accumulated sums. Lastly, we discuss which operations can be put offline to save online evaluator time. We discuss all these in more detail in Section 5.3.

3 Preliminaries

3.1 Notation

Let λ denote the security parameter. For a positive integer m, write $[m] = \{1, \ldots, m\}$. We write $\vec{x}$ for vectors and $\mathbf{1}$ for the all-ones vector. All logarithms are base two unless written as ln, which denotes the natural logarithm. Unless stated otherwise, running time is measured in bit operations, and integer counts are represented in binary. Table 2 summarizes the main parameters.

[5] Such binomial draws can be completed in polynomial time [BKP+14]. See also Section 5.

Fixed-point rounding. For a real number z and a nonnegative integer precision P, write

$$\lfloor z \rfloor_P = 2^{-P}\lfloor 2^P z \rfloor, \qquad \lceil z \rceil_P = 2^{-P}\lceil 2^P z \rceil, \qquad \lfloor z \rceil_P = 2^{-P}\lfloor 2^P z + 1/2 \rfloor.$$

These denote downward, upward, and nearest rounding to P fractional binary digits. A tilde denotes a fixed-point-rounded approximation (e.g., $\tilde{z}$).

Floating-point rounding. Sometimes, we use floating-point rounding to preserve relative precision even for very small or very large values. Specifically, we store the exponent separately and truncate the significand to a fixed number of binary digits.

For $z \neq 0$, write $|z| = m2^e$ with $1 \leq m < 2$ and $e \in \mathbb{Z}$. We truncate the significand m to j fractional binary digits:

$$\mathrm{fl}_j(z) = \mathrm{sgn}(z)\, 2^e \lfloor m \rfloor_j, \qquad \mathrm{fl}_j(0) = 0.$$

The significand and exponent are stored separately, with the exponent written in binary. This gives

$$|\mathrm{fl}_j(z) - z| \leq 2^{-j}|z|, \qquad |\mathrm{fl}_j(z)| \leq |z|.$$

A tilde with fl superscript denotes a floating-point-rounded approximation (e.g., $\tilde{z}^{\mathsf{fl}}$)

3.2 Arithmetic Circuits

Circuit model. An arithmetic circuit is a directed acyclic graph with variable input nodes $x_1, \ldots, x_n$, binary addition and multiplication gates, and one designated output node, called the root. A node (representing the output of a gate) may feed multiple gates, and both inputs of a gate may refer to the same node. We write $|C|$ for the number of arithmetic gates, excluding input nodes, and A for the number of addition gates. The depth is the maximum number of gates on an input–output path; the addition depth is the maximum number of addition gates on such a path.

3.3 Additively Homomorphic Encryption

Definition 3.1 (Additively Homomorphic Encryption (AHE))**.** Fix a public integer modulus $p \geq 2$, not necessarily prime. An additively homomorphic encryption scheme Π_A over $\mathbb{Z}_p$ consists of four PPT algorithms $(\mathsf{KeyGen}_A, \mathsf{Enc}_A, \mathsf{Dec}_A, \mathsf{Add}_A)$ defined as follows:

- $(\mathsf{pk}_A, \mathsf{sk}_A) \leftarrow \mathsf{KeyGen}_A(1^\lambda, p)$: takes a security parameter λ and a plaintext modulus p; outputs a public key pk_A and a secret key sk_A. The public key includes p.
- $c \leftarrow \mathsf{Enc}_A(\mathsf{pk}_A, m)$: takes the public key pk_A and a message $m \in \mathbb{Z}_p$; outputs a ciphertext c.
- $m \leftarrow \mathsf{Dec}_A(\mathsf{sk}_A, c)$: takes the secret key sk_A and a ciphertext c; outputs its plaintext m.
- $c_{\text{out}} \leftarrow \mathsf{Add}_A(\mathsf{pk}_A, c, c')$: takes the public key pk_A and two ciphertexts under that key; outputs a ciphertext encrypting the sum of their plaintexts modulo p. For simplicity, we may also simply write this as $c_{\text{out}} \leftarrow c + c'$.

The scheme must satisfy the following properties. In each property, generate $(\mathsf{pk}_A, \mathsf{sk}_A) \leftarrow \mathsf{KeyGen}_A(1^\lambda, p)$.

- **Correctness.** For every $m \in \mathbb{Z}_p$,

$$\Pr\big[\mathsf{Dec}_A(\mathsf{sk}_A, \mathsf{Enc}_A(\mathsf{pk}_A, m)) = m\big] \geq 1 - \mathsf{negl}(\lambda).$$

For every pair of valid ciphertexts c, c' under the same key,

$$\Pr\big[\mathsf{Dec}_A(\mathsf{sk}_A, c + c') = \big(\mathsf{Dec}_A(\mathsf{sk}_A, c) + \mathsf{Dec}_A(\mathsf{sk}_A, c')\big) \bmod p\big] = 1.$$

These guarantees hold throughout polynomial-length sequences of homomorphic additions, including computations that reuse the same ciphertext.

- **Compactness.** For every fresh or evaluated ciphertext c,

$$|c| \le \mathsf{poly}(\lambda, \log p),$$
$$\mathrm{Time}\big(\mathsf{Dec}_A(\mathsf{sk}_A, c)\big) \le \mathsf{poly}(\lambda, \log p),$$

  independently of the number of homomorphic additions performed. Here $|c|$ denotes ciphertext bit length, and Time denotes worst-case running time.

- **IND-CPA security.** For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, let

$$(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pk}_A), \qquad m_0, m_1 \in \mathbb{Z}_p.$$

  Sample a uniform bit $b \in \{0, 1\}$, generate

$$c^\star \leftarrow \mathsf{Enc}_A(\mathsf{pk}_A, m_b), \qquad b' \leftarrow \mathcal{A}_2(\mathsf{st}, c^\star),$$

  and require

$$\left|\Pr[b' = b] - \frac{1}{2}\right| \le \mathrm{negl}(\lambda).$$

Our main construction uses $p = 2$, in which case plaintext addition is XOR.

Remark 3.2 (Weak additively homomorphic encryption)**.** Some AHE are only "weakly additive", which means that it only allows addition up to a certain depth. Lattice-based AHE schemes (e.g., based on LWE or RLWE) are all of such kind: Each ciphertext contains some initial error and additions among ciphertexts enlarge such errors; when errors are grown beyond certain threshold, the correctness guarantee no longer holds. Instead, such schemes require that the total number of or the depth of the additions is known. However, such weakly additive homomorphism is already sufficient for us: our HIE is leveled HE, thereby requires the level to be known in advance, which can be used to generate the parameters for the underlying AHE to allow correctness holds over the number of additions we need.

3.4 Probability Tools

Binomial counts. For an integer $N \ge 0$ and a probability $p \in [0, 1]$, $\mathrm{Bin}(N, p)$ denotes the number of successes in N independent Bernoulli trials with success probability p.

Concentration of averages. For independent random variables $Y_1, \ldots, Y_K \in [-1, 1]$, Hoeffding's inequality gives, for every $t > 0$,

$$\Pr\left[\left|\frac{1}{K}\sum_{j=1}^{K}(Y_j - \mathbb{E}[Y_j])\right| \ge t\right] \le 2\exp(-Kt^2/2).$$

Statistical distance. For discrete distributions P and Q on the same space, their total-variation distance is

$$d_{\mathrm{TV}}(P,Q) = \frac{1}{2}\sum_{z} |P(z) - Q(z)|.$$

It equals the maximum difference between their probabilities of the same event. Replacing at most T sampling calls by approximations of respective conditional distances $\tau_1, \ldots, \tau_T$ changes the full experiment by distance at most $\sum_{j=1}^{T} \tau_j$. For adaptive calls, each bound is required whenever the preceding histories agree.

4 Definition

Definition 4.1 (Range-Relatively Correct Leveled Homomorphic Encryption)**.** Let $\mathcal{C}_d$ be a family of arithmetic circuits with depth at most d. Let $\mathcal{X}_d$ and $\mathcal{Y}_d$ be the bounded numerical input and output domains, respectively, such that every n-input circuit $C \in \mathcal{C}_d$ maps $\mathcal{X}_d^n$ into $\mathcal{Y}_d$. Real-valued inputs and accuracy parameters have finite rational representations. Let $r = \sup_{x \in \mathcal{X}_d} |x| > 0 \in \mathbb{Q}$ be the chosen public input bound, with a bit length L_r. For each n-input circuit $C \in \mathcal{C}_d$, define its output scale by

$$\eta_C = \sup_{\mathbf{x} \in \mathcal{X}_d^n} |C(\mathbf{x})|.$$

A range-relatively correct leveled homomorphic encryption scheme Π consists of five algorithms $(\mathsf{GenParam}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ defined as follows:

- $\mathsf{pp} \leftarrow \mathsf{GenParam}(1^\lambda, d, r, \varepsilon, \delta)$: takes a security parameter λ, depth bound $d \geq 0$, a public input bound $r > 0 \in \mathbb{Q}$, a relative error tolerance $\varepsilon > 0$, and a failure probability $\delta \in (0,1)$; outputs public parameters pp fixing these values and the supported circuit family and numerical domains.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$: takes the public parameters pp; outputs a public key pk and a secret key sk.
- $c \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, x)$: takes the public parameters pp, public key pk, and an input $x \in \mathcal{X}_d$; outputs an input ciphertext c.
- $c_{\mathrm{out}} \leftarrow \mathsf{Eval}(\mathsf{pp}, \mathsf{pk}, C, c_1, \ldots, c_n)$: takes the public parameters pp, public key pk, a circuit $C \in \mathcal{C}_d$, and its encrypted inputs; outputs an evaluated ciphertext c_{out}.
- $\mathsf{res} \leftarrow \mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, c)$: takes the public parameters pp, secret key sk, and a fresh or evaluated ciphertext c; outputs a numerical value res.

All algorithms run in time polynomial in $\lambda, d, L_r, \varepsilon^{-1}, \log(1/\delta)$ and their input lengths. For cryptographic parameter families, we require $d, L_r, \varepsilon^{-1}, \log(1/\delta)$ and the circuit and real-valued input description lengths to be polynomially bounded in λ.

The scheme must satisfy the following properties. In each property, generate $\mathsf{pp} \leftarrow \mathsf{GenParam}(1^\lambda, d, r, \varepsilon, \delta)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$.

- **Range-relative correctness.** For correctness, instead of using an absolute error, we define a relative error: The output error $\varepsilon \cdot \eta_C$ is scaled with respect to η_C: the maximum magnitude of the circuit output over the input range. Similarly, the input error is $\varepsilon \cdot r$ scaled by r, the maximum input range. This provides a stronger correctness guarantee for circuits with smaller outputs and inputs. More rigorously:

For every circuit $C \in \mathcal{C}_d$ and every input vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{X}_d^n$, let

$$c_i \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, x_i), \qquad i \in [n],$$

and

$$\mathsf{res} \leftarrow \mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{Eval}(\mathsf{pp}, \mathsf{pk}, C, c_1, \ldots, c_n)).$$

It holds that

$$\Pr\left[|\mathsf{res} - C(\mathbf{x})| > \varepsilon \cdot \eta_C\right] \leq \delta + \mathsf{negl}(\lambda).$$

The probability is over all randomness of the algorithms, with C and $\mathbf{x}$ fixed.[6]

- **Compactness.** For every $C \in \mathcal{C}_d$ and every valid input vector, the resulting ciphertexts satisfy

$$\begin{aligned} \max_{i \in [n]} |c_i| &\leq \mathsf{poly}(\lambda, d, L_r, \varepsilon^{-1}, \log(1/\delta)), \\ |c_{\mathrm{out}}| &\leq \mathsf{poly}(\lambda, d, L_r, \varepsilon^{-1}, \log(1/\delta)), \\ \max_{c \in \{c_1, \ldots, c_n, c_{\mathrm{out}}\}} \mathrm{Time}(\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, c)) &\leq \mathsf{poly}(\lambda, d, L_r, \varepsilon^{-1}, \log(1/\delta)). \end{aligned}$$

  Here $|c|$ denotes ciphertext bit length, and Time denotes worst-case running time.

- **IND-CPA security.** For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, let

$$(x_0, x_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pp}, \mathsf{pk}), \qquad x_0, x_1 \in \mathcal{X}_d.$$

  Sample a uniform bit $b \in \{0, 1\}$, generate

$$c^\star \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, x_b), \qquad b' \leftarrow \mathcal{A}_2(\mathsf{st}, c^\star),$$

  and require

$$\left| \Pr[b' = b] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda).$$

Remark 4.2 (Relation to exact and approximate HE.)**.** Since we introduced an alternative correctness definition, we briefly discuss its relationship to HE with exact and approximate correctness. Exact HE (e.g., BGV [BGV14] and BFV [Bra12, FV12]) recovers the prescribed plaintext result in its stated domain, subject to its allowed failure probability. On the other hand, both CKKS [CKKS17] and our scheme instead allow numerical error. CKKS tracks a plaintext-magnitude bound ν and an error bound B, and defines relative error as B/ν. Our range-relative correctness, instead, is with respect to the maximum output magnitude η_C (which can be computed as $C(\vec{r})$ but could also potentially be bounded more tightly; see Section 5.3.1 for details), being looser than CKKS' guarantee. Furthermore, in Remark 5.2, we discuss the implication of this correctness (together with compactness) over the constructions.

Remark 4.3 (Decryption over inputs)**.** Note that our correctness only discusses the correctness after evaluation. However, a normal encryption scheme would instead need to allow decryption for inputs without evaluation as well. Thus, for completeness, we include this definition as well: for the same quantifiers above, fresh ciphertexts satisfy that for every $x \in \mathcal{X}_d$,

$$\Pr\left[\left|\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, x)) - x\right| > \varepsilon \cdot r\right] \leq \mathsf{negl}(\lambda).$$

However, since this is not essential to the main point of the paper, we skip the details.

[6] For completeness, see Remark 4.3 for correctness when decrypting fresh input ciphertexts.

5 Main Construction

We now give the full construction of Homomorphism in Expectation (HIE). We first explain the construction with exact arithmetic and sampling (i.e., an ideal model with no precision loss when handling real numbers), then give its finite-precision instantiation (i.e., the real-world model).

5.1 The Ideal Construction

Let $\Pi_A = (\mathsf{KeyGen}_A, \mathsf{Enc}_A, \mathsf{Dec}_A, \mathsf{Add}_A)$ be an additively homomorphic encryption scheme over $\mathbb{Z}_2$, as in Definition 3.1. The supported family $\mathcal{C}_d$ consists of circuits over $\{+, \times\}$ with total depth at most d. Choose any public input bound $r > 0 \in \mathbb{Q}$, and let L_r denote its bit length. For $d \geq 0$, $0 < \varepsilon \leq 1$, and $0 < \delta < 1/2$, set

$$\ell_{\mathsf{in}} = d + \lceil \log_2(64/\varepsilon) \rceil + 1. \tag{1}$$

The input domain is $\mathcal{X}_d = [-r, r]$. Set $B_{d,r} = 2^{(L_r+1)2^d}$ and $\mathcal{Y}_d = [-B_{d,r}, B_{d,r}]$. Write $\eta_v = C_v(r, \ldots, r)$ for a node's public weight and η_C for the root weight, which is the maximum output magnitude over the input domain. Also set

$$\ell_{\mathsf{out}} = \lceil 16\varepsilon^{-2} \lceil \log_2(8/\delta) \rceil \rceil. \tag{2}$$

See also Table 2 for the explanation of each variable and Section 3.1 for notation.

Ideal procedures. We use three ideal procedures (running in polynomial time), whose finite-precision realizations appear in Section 5.2. The public parameters are implicit in the following routines.

- $\mathsf{IdealWeightPass}(C)$: Recall that we need to compute a weight for each node in the circuit by setting every input to the public value r. We use $\mathsf{IdealWeightPass}(C)$ to compute these weights in circuit C. The output contains (1) p_v, the probability of selecting the left child at each addition node $v = L + R$, given by $p_v = \eta_L/(\eta_L + \eta_R)$; and (2) η_C, the public scale of the circuit, used for decoding.
- $\mathsf{IdealBinSample}(N, p)$: Recall that a gate may receive N requests for independent samples. At an addition gate, each request independently selects the left child with probability p and the right child otherwise. We use $\mathsf{IdealBinSample}(N, p)$ to sample the total number of requests sent to the left child without processing them individually. It outputs a count $k \in \{0, \ldots, N\}$ distributed exactly as $\mathrm{Bin}(N, p)$.
- $\mathsf{IdealDigitCounts}(k_i)$: Recall that for each input x_i, we record the number k_i of independent samples needed for the final product. Each sample selects an encrypted probability digit: digit j is selected with probability 2^{-j}, and the encrypted complementary bit σ_i is selected with the remaining probability $2^{-\ell_{\mathsf{in}}}$. We use $\mathsf{IdealDigitCounts}(k_i)$ to generate these selection counts directly. It outputs $(k_{i,0}, k_{i,1}, \ldots, k_{i,\ell_{\mathsf{in}}})$, where $k_{i,0}$ counts complementary-bit selections and $k_{i,j}$ counts selections of digit j. The output has exactly the same joint distribution as the counts obtained by sampling individually. This procedure can be realized with at most ℓ_{in} calls to $\mathsf{IdealBinSample}$.

Theorem 5.1 (Ideal construction)**.** *Assume that Π_A is correct, compact, and IND-CPA secure. In the exact-arithmetic and exact-sampling model, the construction in Algorithm 1 is compact, preserves input privacy, and evaluates every $C \in \mathcal{C}_d$ on input vector $\vec{x} \in [-r, r]^n$ for some $r > 0 \in \mathbb{Q}$, returning $\mathsf{res} \in [-\eta_C, \eta_C]$ with*

$$\Pr[|\mathsf{res} - C(\vec{x})| > \varepsilon \cdot \eta_C] \leq \delta + \mathsf{negl}(\lambda).$$

Algorithm 1 Ideal HIE with exact arithmetic and sampling

procedure $\mathsf{GenParam}(1^\lambda, d, r, \varepsilon, \delta)$
  Compute (1); fix $\mathcal{X}_d$, $\mathcal{Y}_d$, and $\mathcal{C}_d$ as above.
  return $\mathsf{pp} = (\lambda, d, \varepsilon, \delta, r, \ell_{\mathsf{in}})$ and the domain/family descriptions.
end procedure

procedure $\mathsf{KeyGen}(\mathsf{pp})$
  return $\mathsf{KeyGen}_A(1^\lambda, 2)$.
end procedure

procedure $\mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, x)$
  $u \leftarrow x/r$; $q \leftarrow (1-u)/2$.
  If $q < 1$, set $\sigma = 0$ and use the first ℓ_{in} probability digits of q as $b_1, \dots, b_{\ell_{\mathsf{in}}}$.
  If $q = 1$, set $\sigma = 1$ and $b_1 = \cdots = b_{\ell_{\mathsf{in}}} = 1$.
  return $(\mathsf{in}, \mathsf{Enc}_A(\mathsf{pk}, \sigma), \mathsf{Enc}_A(\mathsf{pk}, b_1), \dots, \mathsf{Enc}_A(\mathsf{pk}, b_{\ell_{\mathsf{in}}}))$.
end procedure

procedure $\mathsf{Eval}(\mathsf{pp}, \mathsf{pk}, C, c_1, \dots, c_n)$
  $(p, \eta_C) \leftarrow \mathsf{IdealWeightPass}(C)$; compute ℓ_{out} by (2).
  for $h = 1, \dots, \ell_{\mathsf{out}}$ **do**
    $N_v \leftarrow 0$ for every node v; $N_{\mathrm{root}} \leftarrow 1$. ▷ N_v is the counter for node v.
    for each non-input node v with $N_v > 0$, parents before children **do**
      if $v = L \times R$ **then**
        $N_L \leftarrow N_L + N_v$; $N_R \leftarrow N_R + N_v$.
      else ▷ $v = L + R$
        $J \leftarrow \mathsf{IdealBinSample}(N_v, p_v)$.
        $N_L \leftarrow N_L + J$; $N_R \leftarrow N_R + (N_v - J)$.
      end if
    end for
    $z_h \leftarrow \mathsf{Enc}_A(\mathsf{pk}, 0)$.
    for $i = 1, \dots, n$ **do**
      $k_i \leftarrow \sum_{v \text{ a leaf labeled } x_i} N_v$.
      $(k_{i,0}, k_{i,1}, \dots, k_{i,\ell_{\mathsf{in}}}) \leftarrow \mathsf{IdealDigitCounts}(k_i)$.
      if $k_{i,0}$ is odd **then**
        $z_h \leftarrow \mathsf{Add}_A(\mathsf{pk}, z_h, c_{i,0})$. ▷ $c_{i,0}$ encrypts the complementary bit σ_i.
      end if
      for each $j \in [\ell_{\mathsf{in}}]$ with $k_{i,j}$ odd **do**
        $z_h \leftarrow \mathsf{Add}_A(\mathsf{pk}, z_h, c_{i,j})$. ▷ $c_{i,j}$ encrypts the j-th probability bit for input i.
      end for
    end for
  end for
  return $(\mathsf{out}, \eta_C, z_1, \dots, z_{\ell_{\mathsf{out}}})$.
end procedure

procedure $\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, c)$
  if $c = (\mathsf{in}, c_0, c_1, \dots, c_{\ell_{\mathsf{in}}})$ **then**
    $\sigma \leftarrow \mathsf{Dec}_A(\mathsf{sk}, c_0)$; $b_j \leftarrow \mathsf{Dec}_A(\mathsf{sk}, c_j)$ for $j \in [\ell_{\mathsf{in}}]$.
    return $r\big(1 - 2(\sum_{j=1}^{\ell_{\mathsf{in}}} b_j 2^{-j} + \sigma 2^{-\ell_{\mathsf{in}}})\big)$.
  else ▷ $c = (\mathsf{out}, \eta_C, z_1, \dots, z_{\ell_{\mathsf{out}}})$
    return $\dfrac{\eta_C}{\ell_{\mathsf{out}}} \displaystyle\sum_{h=1}^{\ell_{\mathsf{out}}} (-1)^{\mathsf{Dec}_A(\mathsf{sk}, z_h)}$.
  end if
end procedure

Proof. Fix $C \in \mathcal{C}_d$ and $\vec{x} \in [-r, r]^n$. As usual, circuit and input descriptions, d, L_r, ε^{-1}, and $\log(1/\delta)$ are polynomially bounded in λ. We first work with the intended plaintext bits.

(1) Range-relative correctness. We first analyze individual sampling with no rounding, then show that collecting counts leaves its distribution unchanged. Lastly, we account for the finite input representation and the finite output average.

(a) Exact evaluation, one sampling request at a time. In this step, we use the original inputs $\vec{x}$ with exact probabilities to represent the input and perform sampling one at a time; collecting sampling requests and rounding inputs are handled in (b) and (c). For each node v, let $u_v = C_v(\vec{x})/\eta_v$ be its normalized value. The public weights satisfy

$$0 < \eta_v = C_v(r, \ldots, r), \qquad |C_v(\vec{x})| \le \eta_v.$$

Thus $u_v \in [-1, 1]$ regardless of the magnitude of η_v.

A single request at v returns a bit $B_v \in \{0, 1\}$ and its corresponding stream bit is $S_v = 1 - 2B_v \in \{-1, 1\}$. At an input, we sample the bit directly. At an addition, we select one child and sample the bit according to the encrypted probability. At a multiplication, we sample both children and XOR their bits. Every sampling uses fresh randomness. We prove by induction below that

$$B_v \sim \text{Bernoulli}\left(\frac{1 - u_v}{2}\right), \qquad \mathbb{E}[S_v] = u_v = \frac{C_v(\vec{x})}{\eta_v}. \tag{3}$$

At an input $v = x_i$, we have $\eta_v = r$ and sample $B_v \sim \text{Bernoulli}((1 - x_i/r)/2)$, proving the base case. For an internal node, assume the claimed Bernoulli law holds for each child occurrence. Write $q_L = (1 - u_L)/2$ and $q_R = (1 - u_R)/2$.

At an addition $v = L + R$, select the left child with probability $p_v = \eta_L/(\eta_L + \eta_R)$ and the right child otherwise, independently of the child's sampling randomness. Return the selected child's bit. Since $\eta_v = \eta_L + \eta_R$,

$$\Pr[B_v = 1] = p_v q_L + (1 - p_v) q_R = \frac{1 - (p_v u_L + (1 - p_v) u_R)}{2} = \frac{1 - u_v}{2}.$$

At a multiplication $v = L \times R$, sample two independent child bits and return $B_v = B_L \oplus B_R$. By the induction hypothesis, $B_L \sim \text{Bernoulli}(q_L)$ and $B_R \sim \text{Bernoulli}(q_R)$. Since $\eta_v = \eta_L \eta_R$ and $u_v = u_L u_R$, its Bernoulli parameter is exactly

$$\begin{aligned} p'' = \Pr[B_v = 1] &= q_L(1 - q_R) + (1 - q_L) q_R \\ &= \frac{(1 - u_L)(1 + u_R) + (1 + u_L)(1 - u_R)}{4} \\ &= \frac{1 - u_L u_R}{2} = \frac{1 - u_v}{2}. \end{aligned}$$

Equivalently, $S_v = S_L S_R$. Note that sampling randomness independence here is between sampled occurrences, not between circuit nodes: if $L = R$, we still make two independent calls rather than reuse one bit. This completes the induction.

The root bit is the XOR of all sampled input-occurrence bits. Starting at the root gives $B_{\text{root}} \sim \text{Bernoulli}(\frac{1 - C(\vec{x})/\eta_C}{2})$. Repeating the entire experiment with fresh, independent randomness produces i.i.d. root bits, hence i.i.d. signed samples S_h with $\mathbb{E}[\eta_C \cdot S_h] = C(\vec{x})$.

(b) Collecting requests does not change their distribution. We collect all requests reaching a shared node by processing its parents first. At an addition node v with count N_v, the number of independent choices going left is exactly $\text{Bin}(N_v, p_v)$. Multiplication forwards all N_v requests to both child edges. These are precisely the count updates in Algorithm 1.

The same reasoning applies to input sampling: reordering the stream bits does not change their product. When samples are generated from probability digits, $\mathsf{IdealDigitCounts}$ returns exactly the joint counts of independent selections of the probability digits and complementary bit. Their parities give the same final XOR, including $c_{i,0}$ precisely when $k_{i,0}$ is odd. This preserves the distribution exactly.

Next, we account for the rounded input probability.

(c) Input rounding introduces only a small error. Let $\tilde{q}_i = \sum_{j=1}^{\ell_{\mathsf{in}}} b_{i,j} 2^{-j} + \sigma_i 2^{-\ell_{\mathsf{in}}}$ be the probability represented by the encrypted digits and complementary bit, and set $\tilde{x}_i = r(1 - 2\tilde{q}_i)$. Selecting digit j with probability 2^{-j}, or σ_i with the remaining probability $2^{-\ell_{\mathsf{in}}}$, produces a bit equal to one with probability exactly $\tilde{q}_i$. Here $\sigma_i = 1$ if $q_1 = 1$ and $\sigma_i = 0$ otherwise: when $q_i < 1$, it is zero and the digits give the usual downward truncation; when $q_i = 1$, all digits and σ_i are one, so $\tilde{q}_i = 1$ exactly. In particular, both endpoints $x_i = \pm r$ are represented exactly, and $\tilde{x}_i \in [-r, r]$. Thus steps (a)–(b), applied to $\tilde{\vec{x}}$, show that the implemented output stream bit has mean $C(\tilde{\vec{x}})/\eta_C$.

It remains to compare this target with $C(\vec{x})$. The downward truncation for $q_i < 1$ and exact representation for $q_i = 1$ give $0 \le q_i - \tilde{q}_i < 2^{-\ell_{\mathsf{in}}}$, hence $|\tilde{x}_i - x_i|/r < 2^{1-\ell_{\mathsf{in}}}$. After dividing each node value by its public weight, addition is a convex combination and does not increase the maximum child error. At multiplication, values lie in $[-1, 1]$ and $|yz - y'z'| \le |y - y'| + |z - z'|$. The normalized error therefore at most doubles per level, giving

$$|C(\tilde{\vec{x}}) - C(\vec{x})| \le \eta_C 2^{d+1-\ell_{\mathsf{in}}} \le \varepsilon{\cdot}\eta_C/64. \tag{4}$$

(d) A finite output average gives the required accuracy. The ideal decoder performs no numerical rounding. Its remaining approximation is to estimate the mean using ℓ_{out} independent output stream bits. By Hoeffding's inequality and (2),

$$\Pr\left[\left|\frac{\eta_C}{\ell_{\mathsf{out}}}\sum_{h=1}^{\ell_{\mathsf{out}}} S_h - C(\tilde{\vec{x}})\right| > \frac{\varepsilon \cdot \eta_C}{2}\right] \le 2\exp\left(-\frac{\ell_{\mathsf{out}}\varepsilon^2}{8}\right) \le \delta/4.$$

Combining the input error $\varepsilon{\cdot}\eta_C/64$ with the averaging error $\varepsilon{\cdot}\eta_C/2$ proves the claim, up to the negligible base-encryption correctness failure. Fresh decryption has error below $r2^{1-\ell_{\mathsf{in}}} \le \varepsilon r/64$. Evaluated decryption returns a value in $[-\eta_C, \eta_C] \subseteq \mathcal{Y}_d$.

(2) Compactness. Now we discuss compactness for input and output sizes and decryption time.

(a) Input compactness: each input contains $\ell_{\mathsf{in}} + 1$ compact base ciphertexts, determined by the declared parameters.

(b) Output compactness: each output contains ℓ_{out} base ciphertexts and one public numerical value. Here

$$\ell_{\mathsf{out}} = \lceil 16\varepsilon^{-2}\lceil \log_2(8/\delta)\rceil\rceil.$$

This count is independent of the circuit's depth, size, and number of inputs; the exact scale occupies one slot in the ideal model.

(c) Computation compactness: fresh decryption uses $\ell_{\mathsf{in}} + 1$ base decryptions and $O(\ell_{\mathsf{in}})$ exact arithmetic operations. Evaluated decryption uses ℓ_{out} base decryptions of Π_A and $O(\ell_{\mathsf{out}})$ exact arithmetic operations, accessing only the output stream bits and scale.

(3) Input privacy. The input record consists of encryptions of both its complementary bit and probability digits. A standard hybrid argument over these polynomially many bits reduces input privacy to the IND-CPA security of Π_A. □

Ciphertext counts and ideal computation. Each input contains $\ell_{\mathsf{in}} + 1$ base ciphertexts. Each output contains ℓ_{out} base ciphertexts and its public scale, with ℓ_{out} given by (2). Each counter is at most 2^d, so it occupies only $d + 1$ bits. The public weight pass uses $|C|$ arithmetic operations. Each output repetition then uses $O(|C| + n\ell_{\mathsf{in}})$ counter operations, at most $A + n\ell_{\mathsf{in}}$ binomial draws, and at most $n(\ell_{\mathsf{in}} + 1)$ ciphertext additions.

Remark 5.2 (Non-triviality of HIE)**.** Without compactness, fully homomorphic encryption has a trivial solution: the evaluator returns all encrypted inputs together with the circuit, and the client decrypts the inputs and evaluates the circuit locally. Our range-relative correctness guarantee raises a natural question: could a simpler solution already suffice?

Example circuit families. Indeed, for certain circuit families, this might be possible:

- Addition-only circuits are one such family. The circuit computes a weighted sum $\sum_i w_i x_i$ of its inputs, where w_i counts the occurrences of x_i. With $W = \sum_i w_i$, the output equals $W \cdot \mathbb{E}[x_I]$, where the index I is drawn with probability w_i/W; the output is thus an expectation over a distribution on the inputs, scaled by W. Range-relative correctness therefore reduces the evaluator's task to estimating this expectation, which it does by returning the ciphertexts of k indices drawn from this distribution. The client decrypts them, averages, and multiplies by W; since, by a standard concentration bound, k depends only on ε and δ, the guarantee is met for this family without any homomorphic computation.
- A second family consists of circuits that, with error ε, admit a short description applied to a short portion of the inputs. One example is $C(x) = x^{2^d}$ for $d \geq 1$, evaluated by d successive squarings, with short $x \in [-1, 1]$ and $\eta_C = 1$. For $m = \lceil 2/\varepsilon \rceil$ and $t_j = (j/m)^{1/2^d}$, the table

  $$\mathcal{T} = \{(t_j, j/m) : j = 0, \ldots, m\}$$

  defines the step function that returns j/m, where j is the largest index with $t_j \leq |x|$; this function differs from C by at most $1/m \leq \varepsilon/2$, since $t_j^{2^d} = j/m$. The table has $O(1/\varepsilon)$ entries and depends only on d and ε, so it is a short description of C up to error ε. The evaluator then returns the input ciphertext c together with $\mathcal{T}$, and the client evaluates the description: it decrypts c, reduces x to the summary j, and returns j/m, with no homomorphic computation.

Neither family covers general circuits. Consider

$$C(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{n} (x_i + y_i), \qquad x_i, y_i \in \{0, 1/2\},$$

which has addition depth one and $\eta_C = 1$. First, the output is a product, not a weighted sum, so no small sample estimates it. Second, its output is 1 when every input equals $1/2$ and $1/2$ when a single input is changed to zero, so for $\varepsilon < 1/4$ every correct answer must tell the two cases apart: any description that ignores one coordinate returns the same answer on two inputs whose outputs differ by $1/2$. A correct description must therefore act on all $2n$ inputs. The circuit therefore belongs to neither family.

Precision does not trivialize homomorphic computation. In general, the precision of an answer places no bound on the size or depth of the computation producing it. Even with constant accuracy and confidence, the number of encrypted inputs and the circuit complexity can grow, while compactness still requires the returned answer and decryption work to avoid dependence on the input count and circuit size. Allowing quadratic dependency on the error bound therefore does not permit non-compact solutions that exhaustive enumeration of the inputs, exponential unfolding of the circuit, or transferring the computation back to the client.

5.2 The Finite-Precision Instantiation of the Ideal Construction

Algorithm 2 Finite-precision instantiation of HIE; changes from Algorithm 1 are in blue

procedure $\mathsf{GenParam}(1^\lambda, d, r, \varepsilon, \delta)$
  Compute (1); fix $\mathcal{X}_d$, $\mathcal{Y}_d$, and $\mathcal{C}_d$ as above.
  return $\mathsf{pp} = (\lambda, d, \varepsilon, \delta, r, \ell_{\mathsf{in}})$ and the domain/family descriptions.
end procedure

procedure $\mathsf{KeyGen}(\mathsf{pp})$
  return $\mathsf{KeyGen}_A(1^\lambda, 2)$.
end procedure

procedure $\mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, x)$
  $u \leftarrow x/r$; $q \leftarrow (1-u)/2$.
  If $q < 1$, set $\sigma = 0$ and use the first ℓ_{in} fractional binary digits of q as $b_1, \ldots, b_{\ell_{\mathsf{in}}}$.
  If $q = 1$, set $\sigma = 1$ and $b_1 = \cdots = b_{\ell_{\mathsf{in}}} = 1$.
  return $(\mathsf{in}, \mathsf{Enc}_A(\mathsf{pk}, \sigma), \mathsf{Enc}_A(\mathsf{pk}, b_1), \ldots, \mathsf{Enc}_A(\mathsf{pk}, b_{\ell_{\mathsf{in}}}))$.
end procedure

procedure $\mathsf{Eval}(\mathsf{pp}, \mathsf{pk}, C, c_1, \ldots, c_n)$
  $(\tilde{p}, \tilde{\eta}_C^{\mathsf{fl}}) \leftarrow \mathsf{WeightPass}(C)$; compute ℓ_{out} by (5).
  for $h = 1, \ldots, \ell_{\mathsf{out}}$ **do**
    $N_v \leftarrow 0$ for every node v; $N_{\mathrm{root}} \leftarrow 1$. ▷ N_v is the counter for node v.
    for each non-input node v with $N_v > 0$, parents before children **do**
      if $v = L \times R$ **then**
        $N_L \leftarrow N_L + N_v$; $N_R \leftarrow N_R + N_v$.
      else ▷ $v = L + R$
        $J \leftarrow \mathsf{BinSample}(N_v, \tilde{p}_v)$. ▷ Using [BKP+14]; see also Lemma A.1.
        $N_L \leftarrow N_L + J$; $N_R \leftarrow N_R + (N_v - J)$.
      end if
    end for
    $z_h \leftarrow \mathsf{Enc}_A(\mathsf{pk}, 0)$.
    for $i = 1, \ldots, n$ **do**
      $k_i \leftarrow \sum_{v \text{ a leaf labeled } x_i} N_v$.
      $(k_{i,0}, k_{i,1}, \ldots, k_{i,\ell_{\mathsf{in}}}) \leftarrow \mathsf{DigitCounts}(k_i)$.
      if $k_{i,0}$ is odd **then**
        $z_h \leftarrow \mathsf{Add}_A(\mathsf{pk}, z_h, c_{i,0})$. ▷ $c_{i,0}$ encrypts the complementary bit σ_i.
      end if
      for each $j \in [\ell_{\mathsf{in}}]$ with $k_{i,j}$ odd **do**
        $z_h \leftarrow \mathsf{Add}_A(\mathsf{pk}, z_h, c_{i,j})$. ▷ $c_{i,j}$ encrypts the j-th probability bit for input i.
      end for
    end for
  end for
  return $(\mathsf{out}, \tilde{\eta}_C^{\mathsf{fl}}, z_1, \ldots, z_{\ell_{\mathsf{out}}})$.
end procedure

procedure $\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, c)$
  if $c = (\mathsf{in}, c_0, c_1, \ldots, c_{\ell_{\mathsf{in}}})$ **then**
    $\sigma \leftarrow \mathsf{Dec}_A(\mathsf{sk}, c_0)$; $b_j \leftarrow \mathsf{Dec}_A(\mathsf{sk}, c_j)$ for $j \in [\ell_{\mathsf{in}}]$.
    return $r\big(1 - 2(\sum_{j=1}^{\ell_{\mathsf{in}}} b_j 2^{-j} + \sigma 2^{-\ell_{\mathsf{in}}})\big)$.
  else ▷ $c = (\mathsf{out}, \tilde{\eta}_C^{\mathsf{fl}}, z_1, \ldots, z_{\ell_{\mathsf{out}}})$
    return $\mathrm{fl}_{\lambda+\ell_{\mathsf{in}}+4}\left(\dfrac{\tilde{\eta}_C^{\mathsf{fl}}}{\ell_{\mathsf{out}}} \displaystyle\sum_{h=1}^{\ell_{\mathsf{out}}} (-1)^{\mathsf{Dec}_A(\mathsf{sk}, z_h)}\right)$.
  end if
end procedure

Main point summary. We now realize the ideal procedures with finite-precision algorithms. The point is that in practice, we cannot process real numbers with infinite precision. This section is to show that using finite-precision approximations does not change the resulting guarantees of HIE.

The input encoding stays the same. Input probabilities and child-selection probabilities retain the fixed-point rounding from Section 3.1. Public weights and the decoded answer use floating points instead, so very large values and very small nonzero answers remain compactly representable (see also Section 3.1). Binomial draws have negligible statistical error. For simplicity, we use λ extra bits of precision so that the additional relative numerical error is negligible. This can be further tightened.

Realizing the procedures. We realize the three ideal algorithms in Section 5.1:

- $\mathsf{WeightPass}(C) \to (\tilde{p}, \tilde{\eta}_C^{\mathsf{fl}})$ realizes $\mathsf{IdealWeightPass}(C) \to (p, \eta_C)$ using rounded weights, child-selection probabilities $\tilde{p}_v$, and decoding scale $\tilde{\eta}_C^{\mathsf{fl}}$. Its pseudocode is given in Algorithm 3.
- $\mathsf{BinSample}(N, p) \to k$ realizes $\mathsf{IdealBinSample}(N, p) \to k$. For this, we can directly use standard binomial-sampling methods [FCT15], with a polynomial time limit. Lemma A.1 in Section A gives its polynomial bit complexity and statistical error at most $2^{-\lambda}$. For completeness, Section A also gives a conceptually simpler alternative realization of $\mathsf{BinSample}$.
- $\mathsf{DigitCounts}(k_i) \to (k_{i,0}, k_{i,1}, \ldots, k_{i,\ell_{\mathsf{in}}})$ realizes $\mathsf{IdealDigitCounts}(k_i) \to (k_{i,0}, k_{i,1}, \ldots, k_{i,\ell_{\mathsf{in}}})$ using $\mathsf{BinSample}$ to generate the probability-digit counts and retaining the final remaining count as $k_{i,0}$, with pseudocode in Algorithm 3.

The output count is unchanged from (2):

$$\ell_{\mathsf{out}} = \lceil 16\varepsilon^{-2} \lceil \log_2(8/\delta) \rceil \rceil. \tag{5}$$

The decoding scale is the stored root weight, with no further rounding. Changes from the ideal construction (Algorithm 1) are marked in color in Algorithm 2.

Theorem 5.3 (Finite-precision construction)**.** *Assume that Π_A is a correct, compact, and IND-CPA secure additively homomorphic encryption scheme over $\mathbb{Z}_2$. The construction in Algorithm 2 is a compact, IND-CPA secure, range-relatively correct leveled homomorphic encryption scheme for $\mathcal{C}_d$ on input vectors $\vec{x} \in [-r, r]^n$ for some $r > 0 \in \mathbb{Q}$, returning $\mathsf{res} \in [-\eta_C, \eta_C]$ with*

$$\Pr[|\mathsf{res} - C(\vec{x})| > \varepsilon\eta_C] \leq \delta + \mathsf{negl}(\lambda).$$

Proof. We build on Theorem 5.1, under the same polynomial parameter bounds.

(1) Range-relative correctness. The ideal proof already handles input rounding, request aggregation, and sample averaging. We only need to account for (a) rounded weights, (b) rounded child-selection probabilities, (c) rounding the final answer, and approximate binomial sampling. We first analyze the rounding errors, assuming exact sampling and correct base decryption.

(a) Rounded weights preserve the rescaled mean. Let $j = \lambda + \ell_{\mathsf{in}} + 4$ be the weight precision in Algorithm 3. Each operation truncates its significand to j fractional binary digits, losing at most a 2^{-j} fraction of its value. Thus $0 < \tilde{\eta}_v^{\mathsf{fl}} \leq \eta_v$. For a node v of depth h_v,

$$0 \leq \eta_v - \tilde{\eta}_v^{\mathsf{fl}} \leq \eta_v(2^{h_v+1} - 1)2^{-j}. \tag{6}$$

To see this, input weights are initialized as $\tilde{r}^{\mathsf{fl}} = \mathrm{fl}_j(r)$, so $0 \leq r - \tilde{r}^{\mathsf{fl}} \leq r2^{-j}$. This base-case error is present when the chosen public input bound is not exactly representable at precision j. At addition,

the relative error is at most the larger child relative error plus 2^{-j}. At multiplication, it is at most the sum of the two child relative errors plus 2^{-j}. Induction gives the bound above.

Let $S_v^* \in \{-1, 1\}$ be the stream bit obtained using exact child-selection probabilities computed from the rounded weights, with represented inputs $\tilde{\vec{x}}$. Its mean, multiplied by the stored weight, satisfies

$$\left|\tilde{\eta}_v^{\mathsf{fl}} \mathbb{E}[S_v^*] - C_v(\tilde{\vec{x}})\right| \le \eta_v - \tilde{\eta}_v^{\mathsf{fl}}.$$

We argue this as follows: At an input, $\mathbb{E}[S_v^*] = \tilde{x}_i/r$, so $|\tilde{r}^{\mathsf{fl}}\,\mathbb{E}[S_v^*] - \tilde{x}_i| \le r - \tilde{r}^{\mathsf{fl}}$, establishing the base case even when r is rounded.

At multiplication, we account for the error in two steps. First, combine the children using their already rounded weights. Before rounding the current gate's weight, the inherited error is at most

$$\tilde{\eta}_L^{\mathsf{fl}}(\eta_R - \tilde{\eta}_R^{\mathsf{fl}}) + \eta_R(\eta_L - \tilde{\eta}_L^{\mathsf{fl}}) = \eta_L\eta_R - \tilde{\eta}_L^{\mathsf{fl}}\tilde{\eta}_R^{\mathsf{fl}}.$$

Second, round the product $\tilde{\eta}_L^{\mathsf{fl}}\tilde{\eta}_R^{\mathsf{fl}}$ down to $\tilde{\eta}_v^{\mathsf{fl}}$. Since the stream bit has mean in $[-1, 1]$, this changes its rescaled mean by at most $\tilde{\eta}_L^{\mathsf{fl}}\tilde{\eta}_R^{\mathsf{fl}} - \tilde{\eta}_v^{\mathsf{fl}}$. Adding the two contributions gives

$$\left(\eta_L\eta_R - \tilde{\eta}_L^{\mathsf{fl}}\tilde{\eta}_R^{\mathsf{fl}}\right) + \left(\tilde{\eta}_L^{\mathsf{fl}}\tilde{\eta}_R^{\mathsf{fl}} - \tilde{\eta}_v^{\mathsf{fl}}\right) = \eta_v - \tilde{\eta}_v^{\mathsf{fl}}.$$

The same final-rounding argument applies at addition gates. This proves the claimed bound. Since the stream bit has mean in $[-1, 1]$, and by the argument for each gate above, this adds at most the further weight loss to the error. I.e., error introduced by rounded weights changes the final result by at most $|\eta_v - \tilde{\eta}_v^{\mathsf{fl}}|$ shown in Eq. (6). This proves the claimed bound.

(b) Rounding the child-selection probabilities has little effect. Each probability is rounded down to $\lambda + \ell_{\mathsf{in}} + 2$ fractional binary digits, so it changes by less than $2^{-(\lambda+\ell_{\mathsf{in}}+2)}$. A traversal processing requests individually makes at most 2^d addition choices. Using the same independent uniform random numbers for both versions, the probability that any choice differs is at most $2^{d-\ell_{\mathsf{in}}-\lambda-2}$. Since stream bits lie in $\{-1, 1\}$, their means differ by at most twice this probability. Thus, writing S for the root stream bit using rounded probabilities,

$$|\mathbb{E}[S] - \mathbb{E}[S_{\mathrm{root}}^*]| \le 2^{d-\ell_{\mathsf{in}}-\lambda-1}.$$

(c) Rounding the final answer has little effect. The unrounded answer has magnitude at most $\tilde{\eta}_C^{\mathsf{fl}} \le \eta_C$. Truncating its significand to j fractional binary digits therefore changes it by at most $\eta_C 2^{-j}$.

Combining the weight error, the child-selection error multiplied by $\tilde{\eta}_C^{\mathsf{fl}}$, and this final rounding error gives

$$\left|\tilde{\eta}_C^{\mathsf{fl}} \cdot \mathbb{E}[S] - C(\tilde{\vec{x}})\right| + \eta_C 2^{-j} \le \eta_C 2^{d-\ell_{\mathsf{in}}-\lambda} \le \frac{\varepsilon\eta_C}{128}\, 2^{-\lambda}. \tag{7}$$

Adding the input-rounding error from (4) gives

$$\left|\tilde{\eta}_C^{\mathsf{fl}} \cdot \mathbb{E}[S] - C(\vec{x})\right| + \eta_C 2^{-j} \le \varepsilon\eta_C/32. \tag{8}$$

Putting everything together. Since $0 < \tilde{\eta}_C^{\mathsf{fl}} \le \eta_C$, the Hoeffding argument from Theorem 5.1, now using $\tilde{\eta}_C^{\mathsf{fl}}$ and (5), bounds the averaging error by $\varepsilon\cdot\eta_C/2$ except with probability $\delta/4$. Together with (8), the total error is at most $\varepsilon \cdot \eta_C/2 + \varepsilon \cdot \eta_C/32 < \varepsilon \cdot \eta_C$. Fresh decryption is unchanged. Truncation toward zero keeps the evaluated answer in $[-\tilde{\eta}_C^{\mathsf{fl}}, \tilde{\eta}_C^{\mathsf{fl}}] \subseteq [-\eta_C, \eta_C]$.

Finite-precision binomial sampling. Finally, replace the exact binomial draws by BinSample, keeping the same rounded child-selection probabilities. By Lemma A.1, each call changes the distribution

by at most $2^{-\lambda}$. There are at most $\ell_{\mathsf{out}}(A + n\ell_{\mathsf{in}})$ calls, where A is the number of addition gates, so their total effect is negligible. Including base-encryption correctness failures yields

$$\Pr[|\mathsf{res} - C(\vec{x})| > \varepsilon\eta_C] \le \delta + \mathsf{negl}(\lambda). \tag{9}$$

(2) Polynomial running time. By Lemma A.1, each binomial draw takes time polynomial in λ and the bit lengths of its count and probability. Here each count is at most 2^d, and each probability has at most $\lambda + \ell_{\mathsf{in}} + 2$ fractional bits. There are only polynomially many such draws.

For the complexity bounds, let h_v denote the depth of node v. Induction on depth gives

$$\min(1, r)^{2^{h_v}} \le \eta_v \le 2^{2^{h_v}-1} \max(1, r)^{2^{h_v}}.$$

In particular, since $2^{-L_r} \le r \le 2^{L_r}$, $2^{-L_r 2^d} \le \eta_v \le B_{d,r}$ for every node.

The remaining arithmetic uses $j + 1$-bit significands and $O(d + \log(L_r + 1))$-bit exponents: the ideal weight bounds derived above and (6) give $2^{-L_r 2^d - 1} \le \tilde{\eta}_v^{\mathsf{fl}} \le 2^{(L_r+1)2^d}$. Computing $\mathrm{fl}_j(r)$ takes time polynomial in $L_r + j$. Rounded operations on these representations take polynomial time, including floating-point addition, multiplication, and the required rounded ratios at the prescribed precisions; large exponent gaps are handled without performing correspondingly large shifts.

(3) Compactness. Let $\ell_A(\lambda)$ bound the length of a base ciphertext.

(a) Input compactness: input encryption is unchanged, so each input still has $\ell_{\mathsf{in}} + 1$ base ciphertexts, independently of C and n.

(b) Output compactness: The number of base ciphertexts is

$$\ell_{\mathsf{out}} = \lceil 16\varepsilon^{-2} \lceil \log_2(8/\delta) \rceil \rceil = O\Big(\varepsilon^{-2} \log(1/\delta)\Big). \tag{10}$$

This count is independent of C, d, and n. The public scale uses $j + 1$ significand bits and $O(d + \log(L_r + 1))$ exponent bits, for a total of $O(\lambda + d + \log(1/\varepsilon) + \log(L_r + 1))$ bits. Thus total output size depends only on the declared parameters, independently of $|C|$ and n.

(c) Computation compactness: decryption uses $\ell_{\mathsf{in}} + 1$ base decryptions for a fresh record and ℓ_{out} for an evaluated output. Fresh decryption reconstructs the encoded input using $O(L_r + \ell_{\mathsf{in}})$-bit integers, including the public rational factor r. Evaluated decryption sums the stream bits, multiplies by the scale, and rounds. It uses $O(\lambda + d + \log(1/\varepsilon) + \log(L_r + 1) + \log(\ell_{\mathsf{out}} + 1))$-bit significands, exponents, and integer operands; it never expands a large binary exponent into a long integer. Its running time is polynomial in the declared parameters, independently of $|C|$ and n.

(4) IND-CPA security. Input encryption is unchanged, so the hybrid proof of Theorem 5.1 applies. The finite evaluator is polynomial time and uses only public information and encrypted inputs, preserving their computational indistinguishability. □

Communication and computation costs. Including everything, input and output sizes are

$$(\ell_{\mathsf{in}} + 1)\ell_A(\lambda) + O(1), \qquad \ell_{\mathsf{out}}\ell_A(\lambda) + O(\lambda + d + \log(1/\varepsilon) + \log(L_r + 1) + \log(\ell_{\mathsf{out}} + 1))$$

bits, respectively. Evaluation uses one public pass over C, at most $\ell_{\mathsf{out}}(A + n\ell_{\mathsf{in}})$ binomial draws, $\ell_{\mathsf{out}} n(\ell_{\mathsf{in}} + 1)$ base additions, and ℓ_{out} encryptions of zero. It uses $O(\ell_{\mathsf{out}}(|C| + n\ell_{\mathsf{in}}))$ counter and index operations. The binomial draws have polynomial bit cost by Lemma A.1.

5.3 Additional Discussion

This section additionally discusses a tighter bound on η_C with public inputs and negative inputs, and also extending HIE to $\mathbb{Z}_p$ for sufficiently large p.

Algorithm 3 Public weights and probability-digit counts

procedure WeightPass(C)
  $j \leftarrow \lambda + \ell_{\mathsf{in}} + 4$.
  for each node v, inputs before parents **do**
    if v is an input **then**
      $\tilde{\eta}_v^{\mathsf{fl}} \leftarrow \mathrm{fl}_j(r)$. ▷ Round the chosen public input weight.
    else if $v = L \times R$ **then**
      $\tilde{\eta}_v^{\mathsf{fl}} \leftarrow \mathrm{fl}_j\big(\tilde{\eta}_L^{\mathsf{fl}} \tilde{\eta}_R^{\mathsf{fl}}\big)$.
    else ▷ $v = L + R$
      $\tilde{p}_v \leftarrow \big\lfloor \frac{\tilde{\eta}_L^{\mathsf{fl}}}{\tilde{\eta}_L^{\mathsf{fl}} + \tilde{\eta}_R^{\mathsf{fl}}} \big\rfloor_{\lambda+\ell_{\mathsf{in}}+2}$.
      $\tilde{\eta}_v^{\mathsf{fl}} \leftarrow \mathrm{fl}_j\big(\tilde{\eta}_L^{\mathsf{fl}} + \tilde{\eta}_R^{\mathsf{fl}}\big)$.
    end if
  end for
  $\tilde{\eta}_C^{\mathsf{fl}} \leftarrow \tilde{\eta}_{\mathrm{root}}^{\mathsf{fl}}$.
  return $(\tilde{p}, \tilde{\eta}_C^{\mathsf{fl}})$.
end procedure

procedure DigitCounts(k_i)
  $k_{i,0} \leftarrow k_i$; $k_{i,j} \leftarrow 0$ for every $j \in [\ell_{\mathsf{in}}]$.
  for $j = 1, \ldots, \ell_{\mathsf{in}}$ while $k_{i,0} > 0$ **do**
    $k_{i,j} \leftarrow \mathsf{BinSample}(k_{i,0}, 1/2)$.
    $k_{i,0} \leftarrow k_{i,0} - k_{i,j}$.
  end for
  return $(k_{i,0}, k_{i,1}, \ldots, k_{i,\ell_{\mathsf{in}}})$.
end procedure

5.3.1 Tighter Bound on η_C

Our construction uses $\eta_C = C(r, \ldots, r)$ as a public bound on the output range. Incorporating known input values and simplifying the circuit before computing its weights can yield a tighter bound. This improves the accuracy guarantee without increasing the number of output samples.

Public inputs. Suppose that C receives encrypted inputs $\vec{x} \in [-r, r]^m$ and public inputs $\vec{y} = (y_1, \ldots, y_{n-m}) \in [-r, r]^{n-m}$ for total input size n. During the public circuit evaluation to compute η_C, instead of assigning weight r to every input, assign weight $|y_j|$ to each public input y_j and compute $\eta_{C,\vec{y}}$ instead. The resulting root weight is

$$\eta_{C,\vec{y}} := C(r, \ldots, r, |y_1|, \ldots, |y_{n-m}|).$$

Since the circuit uses only addition and multiplication,

$$|C(\vec{x}, \vec{y})| \le \eta_{C,\vec{y}} \le \eta_C.$$

Thus the public values give a potentially much smaller bound on all outputs consistent with the remaining encrypted inputs.

With the same ℓ_{in} and ℓ_{out}, the correctness analysis now gives

$$\Pr\big[|\mathsf{res} - C(\vec{x}, \vec{y})| > \varepsilon \cdot \eta_{C,\vec{y}}\big] \le \delta + \mathsf{negl}(\lambda).$$

For example, consider $C(x_1, x_2, y) = y^{32}(x_1 + x_2)$ with $r = 1/2$. If the public value is $y = 1/4$, then $\eta_C = 2^{-32}$ but $\eta_{C,\vec{y}} = 2^{-64}$. The error bound improves from $\varepsilon \cdot 2^{-32}$ to $\varepsilon \cdot 2^{-64}$ with the same number of input and output ciphertexts.

The evaluator uses these precomputed weights $\eta_{v,\vec{y}}$ for node v with public input $\vec{y}$ to select children at addition gates, and uses $\eta_{C,\vec{y}}$ as the decoding scale. This fully skips the need to encrypt $\vec{y}$ or use our encoding to compute over $\vec{y}$. The cost for these public inputs becomes almost the same as for plaintext evaluation.

Negative inputs. When some public inputs are negative (to allow subtractions), substituting them before taking absolute values can reveal cancellations and further tighten the bound. More precisely, after substituting $\vec{y}$ and collecting terms with the same monomial, write

$$C(\vec{x}, \vec{y}) = \sum_{\alpha} c_\alpha(\vec{y}) \vec{x}^\alpha.$$

where $c_\alpha(\vec{y})$ is the coefficient computed via $\vec{y}$ for the term $\vec{x}^\alpha$, and $\alpha = (\alpha_1, \ldots, \alpha_m)$ is a vector of nonnegative integer exponents, $\vec{x}^\alpha = \prod_{i=1}^m x_i^{\alpha_i}$, and $|\alpha| = \sum_{i=1}^m \alpha_i$ is the monomial's total degree. The sum ranges over the monomials appearing in the polynomial.

Define

$$\eta'_{C,\vec{y}} := \sum_{\alpha} |c_\alpha(\vec{y})| r^{|\alpha|}.$$

By the triangle inequality,

$$|C(\vec{x}, \vec{y})| \leq \eta'_{C,\vec{y}} \leq \eta_{C,\vec{y}}.$$

The first inequality comes from the simple fact that $\vec{x} \in [-r, r]$. The second inequality follows because collecting terms allows their coefficients to cancel before taking absolute values, whereas the original weight computation discards their signs first.

For example, consider $C(x, y_1, y_2) = xy_1 + xy_2$. The original weight computation of $\eta_{C,\vec{y}}$ gives $\eta_{C,\vec{y}} = r(|y_1| + |y_2|)$. Combining the public coefficients gives $C(x, \vec{y}) = x(y_1 + y_2)$, which may cancel out and give the tighter bound

$$\eta'_{C,\vec{y}} = r|y_1 + y_2|.$$

In particular, if $y_1 = -y_2$, the circuit is identically zero.

To exploit this improvement, the evaluator uses the simplified representation and computes its corresponding weights, retaining the known signs of the public coefficients. The resulting correctness guarantee is then relative to $\eta'_{C,\vec{y}}$. Of course, fully expanding the polynomial may be expensive, but the same principle applies to local simplifications: combine public coefficients and cancel terms before computing their weights.

5.3.2 Extending HIE to $\mathbb{Z}_p$

Instead of using AHE over $\mathbb{Z}_2$, we could instead use $\mathbb{Z}_p$ for sufficiently large p: Recall that each output bit is the XOR of a public subset of input bits. We can instead add those bits as ordinary integers and let the client take the parity after decryption. An odd sum gives the stream bit -1, and an even sum gives the stream bit $+1$. Thus integer addition recovers exactly the same output stream bit as over $\mathbb{Z}_2$, provided there is no wraparound over p.

This does not affect the computation procedure: we still compute modulo two over the final selection from each binomial sampling for the input probability digits (and the complementary bit) before additions, so the computation complexity remains unchanged for the evaluator. With this, we simply need $p > \min(n \cdot (\ell_{\mathsf{in}} + 1), 2^d)$, which is quite small and can be satisfied by many non-$\mathbb{Z}_2$ AHE schemes (Section 1.2).

Remark 5.4 (Multiplicatively homomorphic encryption)**.** Our construction also works with multiplicatively homomorphic encryption supporting $\{-1, +1\}$, such as the ElGamal [ElG85] variant by

Couteau, Peters, and Pointcheval [CPP16]. This variant operates over the subgroup $J_N \subseteq \mathbb{Z}_N^*$ of elements with Jacobi symbol $+1$, for a suitably generated RSA modulus N. Instead of encrypting each probability digit or complementary bit $b \in \{0,1\}$, encrypt its corresponding sign $(-1)^b \in J_N$ directly. Homomorphic multiplication then implements encrypted XOR, since

$$(-1)^b(-1)^{b'} = (-1)^{b \oplus b'}.$$

The evaluator uses an encryption of $+1$ as its accumulator and multiplies the selected ciphertexts. Even selection counts still cancel because $(-1)^{2k} = 1$. The public sampling, count collection, and scale computation remain unchanged, and decryption averages the recovered signs using the same public scale. Thus the correctness, compactness, and security arguments carry over.

5.3.3 Moving Public Evaluation Offline

Recall that the public weights η_v, child-selection probabilities p_v, request counts (for each gate), and probability-digit selections (for each input) do *not* depend on the encrypted input values. Once the circuit and public parameters are known, all of these can be computed before the encrypted inputs arrive.

For each output, preprocessing records which encrypted input stream bits will appear in the final XOR process. An input's complementary bit is included when its complementary-bit selection count is odd, and a probability digit is included when its selection count is odd. All even counts cancel (due to XOR). What remains is simply a "public subset" selection of the input ciphertexts.

Online evaluation reads these subsets and XORs the corresponding ciphertexts using Add_A of Π_A. It then attaches the precomputed decoding scale. There is *no* online circuit traversal or binomial sampling. Storing each subset as a bit mask uses at most $\ell_{\mathsf{out}} \cdot n \cdot (\ell_{\mathsf{in}} + 1)$ bits in total, and online evaluation uses at most $\ell_{\mathsf{out}} \cdot n \cdot (\ell_{\mathsf{in}} + 1)$ ciphertext additions. Sparse subsets can even be stored as lists of indices to save storage. The offline phase performs the public work from the original evaluator; the online phase only carries out the resulting encrypted combinations.

Reusing the public choices. For a fixed circuit and compatible parameters, the same subsets can be reused across different encrypted input vectors, even under different keys. The correctness guarantee holds for each fixed input chosen independently of the preprocessing randomness. Reusing the subsets may correlate different answers; it does not give a simultaneous guarantee for all inputs or for inputs chosen after inspecting these subsets.

6 Concluding Remarks

Preliminary implementation and application tests. Preliminary experiments suggest that HIE can offer substantial concrete gains for private inference on certain models. For MNIST, with $\varepsilon \approx 2^{-7}$ and $\delta = 2^{-30}$, our implementation achieves an online evaluator/server time of approximately 2.37 ms per inference and total online communication of approximately 488 KB, with an inference accuracy of 98.24%. Compared with an Orion [EGR25] baseline evaluating the same underlying model (98.27%), these correspond to approximately 74× faster online server computation and 1.88× lower online communication.

Relaxing the error tolerance to $\varepsilon = 2^{-5}$, with the same δ, further reduces the online server time to approximately 0.190 ms and total online communication to approximately 33.4 KB per inference, while achieving 97.64% accuracy. This corresponds to approximately 919× faster online server computation and 27.5× lower online communication than Orion, with an accuracy loss of only 0.63 percentage points relative to Orion.

These results suggest promising practical applications of HIE. However, we would like to note that they remain preliminary, require further validation, and rely on additional concrete optimizations beyond those discussed in this paper (such as tightening the bound of the current proof). Since concrete performance is beyond the scope of this work, we leave the implementation details and a comprehensive experimental evaluation to separate work.

On range-relative correctness and Boolean circuits. One major limitation of HIE is its range-relative correctness. The error is measured relative to η_C, the circuit's maximum output magnitude over the supported input domain, which could be far larger than the actual value of $C(\vec{x})$, unfortunately. For example, suppose we want to evaluate a Boolean circuit: its inputs lie in $\{0,1\}$, so compared with the expected $\{0,1\}$ output, the bound η_C can be exponential in the circuit depth, making it unclear how to realize Boolean circuit evaluation via our construction. An important future direction is to strengthen our correctness guarantee: can the error instead be bounded relative to $|C(\vec{x})|$ for the actual input vector $\vec{x}$? Or be bounded relative to an input bound r similar to CKKS?

An alternative way to resolve this issue is to improve HIE's output dimension. The output dimension currently grows quadratically with ε^{-1}, increasing both communication and evaluation costs. This dependence limits the precision achievable with reasonable efficiency. The natural question is whether we can reduce it to linear in ε^{-1}, or even polynomial in $\log(\varepsilon^{-1})$? The latter could potentially greatly strengthen the correctness guarantee.

Acknowledgment

This work made extensive use of ChatGPT6-Astra to draft the proofs and algorithm pseudocode in Section 5 and Appendix A, the toy example in Section B, and to conduct the literature survey, including suggesting the most closely related prior work [KIU+23]. This work also made extensive use of ChatGPT6-Astra to draft the manuscript and verify the technical claims across the paper. The preliminary implementation results in Section 6 and validation of the reported empirical results were carried out entirely by ChatGPT6-Astra and ChatGPT6.1-Sol.

We thank Jikang Bai, Jingnan He, and Kunpeng Wang for their help with an earlier version of this work. We also thank Dan Boneh, Yijia Chang, Henry Corrigan-Gibbs, Ben Fisch, Alexandra Henzinger, Aayush Jain, Huijia Lin, Jaehyung Kim, Kabir Peshawaria, Eran Tromer, Vinod Vaikuntanathan, and Xiao Wang for their helpful comments and discussion.

References

[BCP03] Emmanuel Bresson, Dario Catalano, and David Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In *Advances in Cryptology—ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 37–54. Springer, 2003.

[Ben94] Josh Benaloh. Dense probabilistic encryption. In *Proceedings of the Workshop on Selected Areas in Cryptography*, pages 120–128, 1994.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography – TCC 2005*, pages 325–341. Springer, 2005.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory*, 6(3):13:1–13:36, 2014.

[BHJL17] Fabrice Benhamouda, Javier Herranz, Marc Joye, and Benoît Libert. Efficient cryptosystems from 2^k-th power residue symbols. *Journal of Cryptology*, 30(2):519–549, 2017.

[BKP+14] Karl Bringmann, Fabian Kuhn, Konstantinos Panagiotou, Ueli Peter, and Henning Thomas. Internal DLA: Efficient simulation of a physical growth model. In *Automata, Languages, and Programming – ICALP 2014, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 247–258. Springer, 2014. https://people.mpi-inf.mpg.de/ kbringma/paper/2014ICALP.pdf, Appendix A.2, Theorem 5.

[BLSD26] Huck Bennett, Zhengnan Lai, and Noah Stephens-Davidowitz. Advanced cryptography from lattice isomorphism—new constructions of IBE and FHE. In *Advances in Cryptology – CRYPTO 2026*, volume 16800 of *Lecture Notes in Computer Science*, pages 446–476. Springer, 2026.

[BMM26] Pedro Branco, Giulio Malavolta, and Zayd Maradni. Fully-homomorphic encryption from lattice isomorphism. In *Theory of Cryptography – TCC 2025*, volume 16268 of *Lecture Notes in Computer Science*, pages 220–252. Springer, 2026.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Advances in Cryptology – CRYPTO 2012*, pages 868–886. Springer, 2012.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *SIAM Journal on Computing*, 43(2):831–871, 2014.

[CF15] Dario Catalano and Dario Fiore. Using linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS 2015)*, pages 1518–1529. ACM, 2015.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.

[CGHTV25] Henry Corrigan-Gibbs, Alexandra Henzinger, Yael Tauman Kalai, and Vinod Vaikuntanathan. Somewhat homomorphic encryption from linear homomorphism and sparse LPN. In *Advances in Cryptology – EUROCRYPT 2025*, volume 15602 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2025.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology – ASIACRYPT 2017, Part I*, pages 409–437. Springer, 2017.

[CL15] Guilhem Castagnos and Fabien Laguillaumie. Linearly homomorphic encryption from DDH. In *Topics in Cryptology—CT-RSA 2015*, volume 9048 of *Lecture Notes in Computer Science*, pages 487–505. Springer, 2015.

[CLT22] Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. Threshold linearly homomorphic encryption on $\mathbb{Z}/2^k\mathbb{Z}$. In *Advances in Cryptology—ASIACRYPT 2022, Part II*, volume 13792 of *Lecture Notes in Computer Science*, pages 99–129. Springer, 2022.

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In *Theory of Cryptography – TCC 2015, Part II*, pages 468–497. Springer, 2015.

[CPP16] Geoffroy Couteau, Thomas Peters, and David Pointcheval. Encryption switching protocols. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 308–338. Springer, 2016.

[CS15] Jung Hee Cheon and Damien Stehlé. Fully homomophic encryption over the integers revisited. In *Advances in Cryptology – EUROCRYPT 2015, Part I*, pages 513–536. Springer, 2015.

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Public Key Cryptography – PKC 2001*, pages 119–136. Springer, 2001.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology – EUROCRYPT 2015, Part I*, pages 617–640. Springer, 2015.

[EGR25] Austin Ebel, Karthik Garimella, and Brandon Reagen. Orion: A fully homomorphic encryption framework for deep learning. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '25, page 734–749, New York, NY, USA, 2025. Association for Computing Machinery.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1985.

[FCT15] Martín Farach-Colton and Meng-Tsung Tsai. Exact sublinear binomial sampling. *Algorithmica*, 73(4):637–651, 2015.

[FLA11] Laurent Fousse, Pascal Lafourcade, and Mohamed Alnuaimi. Benaloh's dense probabilistic encryption revisited. In *Progress in Cryptology—AFRICACRYPT 2011*. Springer, 2011.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012.

[Gai67] Brian R. Gaines. Stochastic computing. In *Proceedings of the April 18–20, 1967, Spring Joint Computer Conference*, volume 30 of *AFIPS '67 (Spring)*, pages 149–156, 1967.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology – CRYPTO 2013, Part I*, pages 75–92. Springer, 2013.

[HLRA22] Hsuan Hsiao, Vincent Lee, Brandon Reagen, and Armin Alaghi. Homomorphically encrypted computation using stochastic encodings. In *Workshop on Negative Results, Opportunities, Perspectives, and Experiences (NOPE 2022)*, 2022. Held in conjunction with ASPLOS 2022, February 28.

[IP07] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In *Theory of Cryptography – TCC 2007*, pages 575–594. Springer, 2007.

[JJ25] Abhishek Jain and Zhengzhong Jin. Sometimes-decryptable homomorphic encryption from sub-exponential DDH. In *Advances in Cryptology – CRYPTO 2025, Part III*, pages 406–439. Springer, 2025.

[JL13] Marc Joye and Benoît Libert. Efficient cryptosystems from 2^k-th power residue symbols. In *Advances in Cryptology—EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 76–92. Springer, 2013.

[JLLS26] Aayush Jain, Huijia Lin, Zeyu Liu, and Sagnik Saha. New techniques for fast and shallow FHE bootstrapping and beyond. In Nadia Heninger and Mike Rosulek, editors, *Advances in Cryptology – CRYPTO 2026, Part II*, volume 16801 of *Lecture Notes in Computer Science*, pages 428–460. Springer, 2026.

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In *Advances in Cryptology – EUROCRYPT 2022, Part I*, pages 670–699. Springer, 2022.

[KIU+23] Ryusuke Koseki, Akira Ito, Rei Ueno, Mehdi Tibouchi, and Naofumi Homma. Homomorphic encryption for stochastic computing. *Journal of Cryptographic Engineering*, 13:251–263, 2023.

[LMK+23] Yongwoo Lee, Daniele Micciancio, Andrey Kim, Rakyong Choi, Maxim Deryabin, Jieun Eom, and Donghoon Yoo. Efficient FHEW bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 227–256. Springer, 2023.

[LMP22] Zeyu Liu, Daniele Micciancio, and Yuriy Polyakov. Large-precision homomorphic sign evaluation using FHEW/TFHE bootstrapping. In *Advances in Cryptology – ASIACRYPT 2022*, 2022.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for Ring-LWE cryptography. In *Advances in Cryptology—EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2013.

[LW23] Zeyu Liu and Yunhao Wang. Amortized functional bootstrapping in less than 7 ms, with $\tilde{O}(1)$ polynomial multiplications. In *Advances in Cryptology – ASIACRYPT 2023*, 2023.

[LW24] Zeyu Liu and Yunhao Wang. Relaxed functional bootstrapping: A new perspective on BGV/BFV bootstrapping. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024*, 2024.

[NS98] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 59–66. ACM, 1998.

[OU98] Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology—EUROCRYPT 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 308–318. Springer, 1998.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT 1999*, pages 223–238. Springer, 1999.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6), 2009.

[RVV24] Seyoon Ragavan, Neekon Vafa, and Vinod Vaikuntanathan. Indistinguishability obfuscation from bilinear maps and LPN variants. In *Theory of Cryptography – TCC 2024, Part IV*, pages 3–36. Springer, 2024.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology – EUROCRYPT 2010*, pages 24–43. Springer, 2010.

[WHS+25] Ruida Wang, Jincheol Ha, Xuan Shen, Xianhui Lu, Chunling Chen, Kunpeng Wang, and Jooyoung Lee. Refined TFHE leveled homomorphic evaluation and its application. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 2309–2323. ACM, 2025.

[WWL+24] Ruida Wang, Yundi Wen, Zhihao Li, Xianhui Lu, Benqiang Wei, Kun Liu, and Kunpeng Wang. Circuit bootstrapping: Faster and smaller. In *Advances in Cryptology – EUROCRYPT 2024*, pages 342–372. Springer Nature Switzerland, 2024.

A Finite Binomial Sampling

We include the binomial sampler we need for our finite-precision construction in Algorithm 2 in Section 5.2. We start with an existing construction in [BKP+14] and state its guarantee in a lemma.

We write $\mathrm{Bin}(N, p)$ for the binomial distribution and $\mathsf{BinSample}(N, p)$ for our sampling algorithm. We build $\mathsf{BinSample}$ using the exact sampler of Bringmann, Kuhn, Panagiotou, Peter, and Thomas [BKP+14, full version, Appendix A.2, Theorem 5] as a black box. Their sampler handles success probability $1/2$; we extend it to general probabilities as follows.

Intuition. We represent p in binary and sample according to its digits, one digit at a time. At each digit, a fair-binomial draw splits the remaining trials into two groups. The digit tells us whether to count one group as successes or discard it; the other group continues to the next digit. We keep only the group sizes, so processing b digits requires at most b fair-binomial draws, regardless of the number of trials.

Algorithm 4 Binomial sampling from fair-binomial draws

Require: Integer $N \geq 0$ and probability $p \in [0, 1]$ with a finite binary representation

if $N = 0$ or $p = 0$ **then**
  return 0
end if
if $p = 1$ **then**
  return N
end if
Write $p = \sum_{j=1}^{b} p_j 2^{-j}$, where $p_j \in \{0, 1\}$
$k \leftarrow 0$, $R \leftarrow N$ ▷ Successes and remaining trials
for $j = 1, \ldots, b$ **do**
  if $R = 0$ **then**
    break
  end if
  $z \leftarrow \mathsf{BinSample}(R, 1/2)$ ▷ See [BKP+14].
  if $z = \perp$ **then**
    return 0
  end if
  if $p_j = 1$ **then**
    $k \leftarrow k + z$, $R \leftarrow R - z$ ▷ Count one group; continue with the other
  else
    $R \leftarrow z$ ▷ Keep one group; discard the other
  end if
end for
return k

Lemma A.1 (Finite binomial sampling)**.** *For an integer $N \geq 0$, a probability $p \in [0, 1]$ represented with b fractional binary digits, and $\lambda \geq 1$, there is a randomized algorithm $\mathsf{BinSample}(N, p)$ (Algorithm 4) returning an integer in $\{0, \ldots, N\}$ in worst-case time $\mathsf{poly}(\lambda, b, \log_2(N + 2))$, measured in bit operations, with*

$$\Delta_{\mathrm{TV}}(\mathsf{BinSample}(N, p), \mathrm{Bin}(N, p)) \leq 2^{-\lambda},$$

where Δ_{TV} denotes total-variation distance.

Proof sketch. Handle $N = 0$ and $p \in \{0, 1\}$ directly. With exact fair-binomial draws, processing the binary digits as above gives the desired binomial distribution; any trials still undecided after the last digit count as failures.

The running-time tail in [BKP+14, full version, Appendix A.2, Theorem 5] lets us cap each draw after polynomially many Word RAM operations with timeout probability at most $2^{-\lambda}/b$. Return zero if any draw times out. A union bound over at most b draws bounds the total timeout probability, and hence the statistical error, by $2^{-\lambda}$. Simulating the bounded word operations gives the claimed polynomial bit complexity. □

B Toy Example

Consider the circuit

$$C(x_1, x_2, x_3) = (x_1 + x_2x_3)x_3 + x_2x_3.$$

Write $g = x_2x_3$ for the shared multiplication gate, so

$$C = (x_1 + g)x_3 + g.$$

Both occurrences of g refer to the same node. The circuit has total depth $d = 4$, and we choose the public input bound $r = 1/4$. We use exact public weights and sampling probabilities below.

1. **Compute the public weights.** Set every input to $r = 1/4$ and evaluate the circuit:

Node	Public weight
$x_1,\ x_2,\ x_3$	$1/4$
$g = x_2x_3$	$1/16$
$x_1 + g$	$5/16$
$(x_1 + g)x_3$	$5/64$
C	$9/64$

   Thus $\eta_C = 9/64$. The evaluator starts each output sample at the root, and the client applies this scale during decoding.

2. **Encode concrete inputs.** Suppose

$$x_1 = \frac{1}{8}, \qquad x_2 = -\frac{1}{8}, \qquad x_3 = \frac{3}{16}.$$

   Their normalized values are $u_i = x_i/r = 4x_i$. For each input, let $q_i = (1 - u_i)/2$ be the probability of sampling -1, and let $\sigma_i = 1$ if $q_i = 1$ and $\sigma_i = 0$ otherwise be its encrypted complementary bit:

Input	u_i	σ_i	q_i	$(b_{i,1}, b_{i,2}, b_{i,3})$
x_1	$1/2$	0	$1/4$	$(0, 1, 0)$
x_2	$-1/2$	0	$3/4$	$(1, 1, 0)$
x_3	$3/4$	0	$1/8$	$(0, 0, 1)$

   The three displayed probability digits represent these inputs exactly; all remaining digits are zero. The client encrypts the probability digits and complementary bits. None of these inputs is the endpoint $u_i = -1$, so all three complementary bits are zero.

3. **Follow the circuit and collect requests.** At the root addition, the evaluator selects $(x_1 + g)x_3$ with probability

$$\frac{5/64}{5/64 + 1/16} = \frac{5}{9},$$

and selects g with probability $4/9$.

Suppose it selects the left branch. The multiplication requests one sample from $x_1 + g$ and one from x_3. The inner addition selects x_1 with probability

$$\frac{1/4}{1/4 + 1/16} = \frac{4}{5},$$

and selects g with probability $1/5$.

Suppose it selects g. We process this shared gate after both of its parents: the root addition and the inner addition. Its counter combines their contributions,

$$N_g = \underbrace{0}_{\text{from the root}} + \underbrace{1}_{\text{from the inner addition}} = 1.$$

The gate $g = x_2x_3$ then sends one request to x_2 and one to x_3. Together with the request already sent to x_3 by the outer multiplication, the input counts are

$$k_1 = 0, \qquad k_2 = 1, \qquad k_3 = 2.$$

Equivalently, the selected input list is

$$[x_2, x_3, x_3].$$

Here the root addition makes the two paths to g mutually exclusive, so N_g never exceeds one within a single output repetition. Nevertheless, both parents contribute to the same counter, and g is stored and processed as one gate.

4. **Sample digit counts and form the encrypted XOR.** We need one sample from x_2 and two independent samples from x_3. Let $k_{i,j}$ count how many requests for input x_i select probability digit j, and let $k_{i,0}$ count complementary-bit selections. All complementary-bit counts are zero in the particular draws below.

For x_2, suppose the digit counts are

$$(k_{2,1}, k_{2,2}, k_{2,3}) = (0, 1, 0).$$

These counts arise by drawing zero from $\mathrm{Bin}(1, 1/2)$ at the first digit and one at the second. The contribution to the final XOR is

$$b_{2,2}.$$

No complementary bit is included because $k_{2,0} = 0$.

For x_3, suppose

$$(k_{3,1}, k_{3,2}, k_{3,3}) = (1, 0, 1).$$

The first draw from $\mathrm{Bin}(2, 1/2)$ returns one. One request remains; the second draw returns zero, and the third returns one. The contribution is

$$b_{3,1} \oplus b_{3,3}.$$

No complementary bit is included because $k_{3,0} = 0$.

Using encrypted additions, the evaluator produces an encryption of

$$z = b_{2,2} \oplus b_{3,1} \oplus b_{3,3}.$$

For the displayed input bits,

$$z = 1 \oplus 0 \oplus 1 = 0.$$

Thus this repetition produces the stream bit $S = (-1)^z = +1$. The sample value changes under the new encoding, but its expectation remains the same.

5. **Check the expectation and decode.** There are three possible selected input lists:

Selected inputs	Probability	Conditional mean
$[x_1, x_3]$	$\frac{5}{9} \cdot \frac{4}{5} = \frac{4}{9}$	$u_1 u_3 = \frac{3}{8}$
$[x_2, x_3, x_3]$	$\frac{5}{9} \cdot \frac{1}{5} = \frac{1}{9}$	$u_2 u_3^2 = -\frac{9}{32}$
$[x_2, x_3]$	$\frac{4}{9}$	$u_2 u_3 = -\frac{3}{8}$

The repeated occurrences of x_3 use independent samples, which is why their product has mean u_3^2.

Combining the cases gives

$$\mathbb{E}[S] = \frac{4}{9} \cdot \frac{3}{8} + \frac{1}{9} \cdot \left(-\frac{9}{32}\right) + \frac{4}{9} \cdot \left(-\frac{3}{8}\right) = -\frac{1}{32}.$$

Multiplying by the public scale recovers the circuit value:

$$\eta_C \, \mathbb{E}[S] = \frac{9}{64} \cdot \left(-\frac{1}{32}\right) = -\frac{9}{2048}.$$

Indeed,

$$C\left(\frac{1}{8}, -\frac{1}{8}, \frac{3}{16}\right) = \left(\frac{1}{8} - \frac{3}{128}\right) \frac{3}{16} - \frac{3}{128} = -\frac{9}{2048}.$$

Repeating independently ℓ_{out} times gives encrypted stream bits $S_1, \ldots, S_{\ell_{\mathsf{out}}}$. The client decrypts them and returns

$$\mathsf{res} = \frac{\eta_C}{\ell_{\mathsf{out}}} \sum_{h=1}^{\ell_{\mathsf{out}}} S_h = \frac{9}{64\ell_{\mathsf{out}}} \sum_{h=1}^{\ell_{\mathsf{out}}} S_h.$$

For target error $\varepsilon\eta_C = 9\varepsilon/64$ and failure probability δ, the ideal construction uses

$$\ell_{\mathsf{out}} = \lceil 16\varepsilon^{-2} \lceil \log_2(8/\delta) \rceil \rceil.$$