FHE-SPEAR: End-to-End Homomorphic RAG with Softmax-Free Generation

Alper-Ender Osman

Abstract

Retrieval-augmented generation under homomorphic encryption requires running both similarity search and language model inference on encrypted data. The generation side is the bottleneck: softmax attention demands high-degree polynomial approximations in CKKS that quickly exhaust the multiplicative depth budget.

Existing encrypted inference systems typically handle classification and single forward passes but not autoregressive generation. FHE-SPEAR replaces softmax with a recurrent SSM whose dominant nonlinearity is a single squaring, packs two embedding dimensions into each CKKS slot to halve ciphertext count, and applies baby-step giant-step (BSGS) diagonal decomposition to cut per-projection rotations by 253×. The result is a complete encrypted RAG pipeline: SIMD-batched retrieval over 50k documents runs in sub-second time, and a client-aided generation protocol with pre-encoded weight diagonals runs the full 24-block, 1.5B-parameter model at 79 seconds per token on an A100 (0.4B at 19 s), producing output identical to plaintext. A fully encrypted FFN mode with CKKS bootstrap removes client interaction from the feedforward path, reaching correlation 0.999989 over 24 blocks. Code: https://github.com/mozendr/FHE-SPEAR.

1 Introduction

Embedding-based similarity search underlies modern retrieval systems. When embeddings encode sensitive information (security vulnerabilities, medical records, proprietary data), computing similarities without revealing them is necessary.

Fully Homomorphic Encryption (FHE) allows computation on encrypted data. The CKKS scheme [1] supports approximate arithmetic on floating-point vectors, making encrypted dot products possible. However, two challenges limit deployment:

1. **Ciphertext size and latency scale with dimension.** Standard 1536-dimensional embeddings require large polynomial modulus degrees, producing ∼326KB ciphertexts and ∼31ms latency per similarity.
2. **Encryption introduces noise.** CKKS operations are approximate, and noise can degrade similarity accuracy.

Dimensionality reduction mitigates both: projecting to fewer dimensions reduces ciphertext size and noise accumulation. But retrieval is only half the problem. Generating text from retrieved context also requires operating on encrypted data, and transformer architectures require expensive polynomial approximations of softmax attention under FHE.

We present FHE-SPEAR, a system for private retrieval-augmented generation under CKKS. For retrieval, we show that SVD projection and SIMD-batched complex packing enable sub-second encrypted similarity search. For generation, we observe that state space models (specifically RWKV-7) avoid softmax entirely, making fully encrypted inference feasible. We combine these into a complete pipeline where the server never observes any plaintext query, document, or generated token. In the client-aided protocol, the client (data owner) decrypts intermediate activations to apply nonlinear operations and re-encrypts before each server-side projection; the server's view is CKKS ciphertexts throughout.

Our contributions:

1. **Complex Packing**: Encoding two dimensions per CKKS slot using real and imaginary components. Queries packed as conjugates yield inner products in the real part. This halves ciphertext count compared to vertical packing.
2. **Encrypted SSM Inference**: Multi-layer RWKV inference under CKKS with CT-CT squaring and no intermediate decryption. BSGS diagonal acceleration extends fully encrypted inference to 19 blocks at d=2048 (perfect correlation) and 24 blocks with CKKS bootstrap (correlation 0.99999). Magnitude-controlled normalization and client-side head completion support 65,536 tokens.
3. **Encrypted Similarity Joins**: CT-CT similarity computation where both queries and documents are encrypted, with SIMD batching and GPU acceleration for throughput.
4. **Hyperbolic Embeddings**: The Lorentz hyperboloid model provides FHE-compatible hyperbolic geometry via the bilinear form $\langle q, d\rangle_{\mathcal{L}} = -q_0 d_0 + \sum_{i>0} q_i d_i$.
5. **FHE-Sim**: Predicts encrypted similarity accuracy from embedding statistics using $\rho_{\text{FHE}} = \rho_{\text{compression}} \times \rho_{\text{noise}}$, validated at 1.7% mean error (Appendix D).

2 Background

2.1 CKKS Homomorphic Encryption

The CKKS scheme [1] encrypts vectors of real numbers with approximate arithmetic:

$$\text{sim}(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y} \approx \text{Dec}(\text{Enc}(\mathbf{x}) \cdot \mathbf{y}) \tag{1}$$

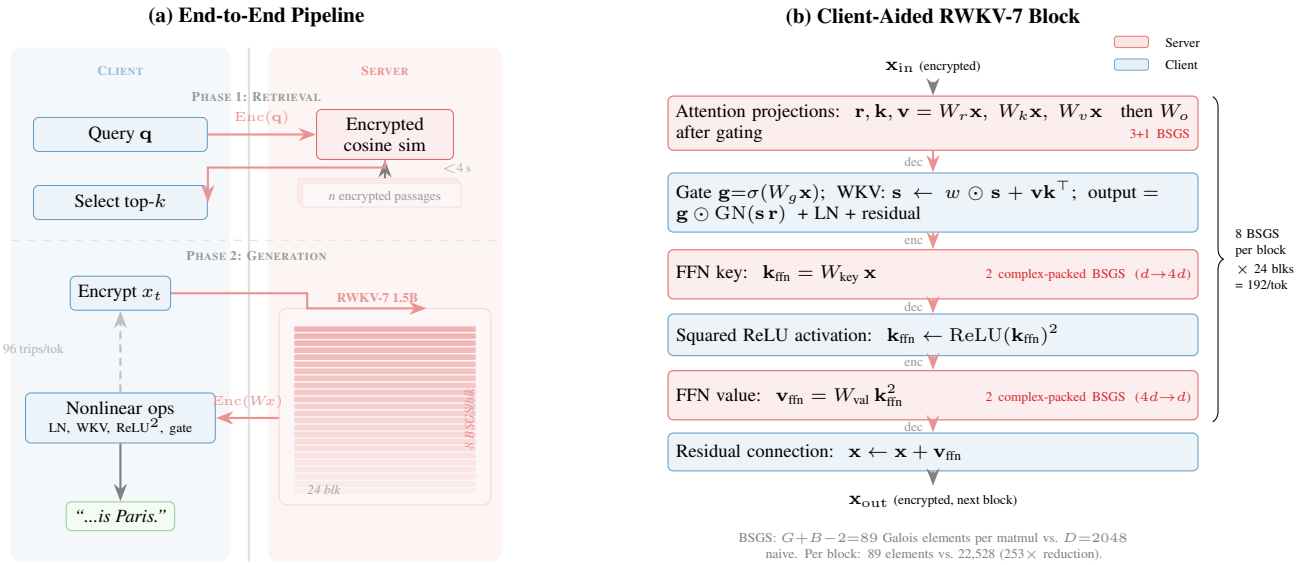

Figure 1: **(a)** Two-phase client-server protocol. Phase 1 retrieves relevant passages via encrypted cosine similarity over the document corpus. Phase 2 generates tokens by running RWKV-7 blocks on the server (BSGS-accelerated diagonal matrix-vector products) with client-side nonlinear operations, requiring 96 round-trips per generated token. The server operates exclusively on CKKS ciphertexts. **(b)** Anatomy of a single RWKV-7 block. The server performs 8 BSGS matrix-vector products per block (4 attention, 2 complex-packed FFN key, 2 complex-packed FFN value); the client handles all nonlinear operations (layer normalization, WKV recurrence, gating, squared ReLU). Data is encrypted and decrypted at each boundary crossing, yielding 4 round-trips per block.

Ciphertext size and operation latency depend on the polynomial modulus degree N:

- $N = 8192$: 326KB ciphertexts, ∼31ms per similarity
- $N = 4096$: 79KB ciphertexts, ∼4ms per similarity

2.2 Problem Formulation

Given embeddings $\{\mathbf{x}_i\}_{i=1}^M$ where $\mathbf{x}_i \in \mathbb{R}^D$, we seek a compression function $f : \mathbb{R}^D \to \mathbb{R}^d$ (with $d \ll D$) maximizing:

$$\rho\Big(\{\mathbf{x}_i^\top \mathbf{x}_j\}, \{\mathrm{FHE}(f(\mathbf{x}_i), f(\mathbf{x}_j))\}\Big) \tag{2}$$

where ρ is Pearson correlation and $\mathrm{FHE}(\cdot,\cdot)$ computes similarity under encryption.

2.3 Threat Model and Protocol

We consider two settings differing in what the server observes:

PT-CT. The server holds plaintext document embeddings $\{d_i\}$. A client encrypts a query $\mathrm{Enc}(q)$; the server computes ciphertext-plaintext products $\mathrm{Enc}(q) \cdot d_i$ and returns encrypted scores. The server learns nothing about query content.

CT-CT (Encrypted Similarity Joins). Both query and document embeddings are encrypted. The server holds $\{\mathrm{Enc}(d_i)\}$ and receives $\mathrm{Enc}(q)$. Using CKKS ciphertext-ciphertext multiplication, the server computes $\mathrm{Enc}(q \cdot d_i)$ without observing either embedding. This provides mutual privacy: neither party reveals their data to the server.

Access pattern hiding (which documents were retrieved) requires ORAM techniques orthogonal to this work.

CKKS Parameters. Retrieval experiments use $N = 4096$ with coefficient modulus $[40, 20, 40]$ bits and scale 2^{20}, achieving ∼128-bit security with 79KB ciphertexts and ∼4ms per similarity. Generation experiments (Section 8) use $N = 32768$ with scale 2^{40} to provide sufficient multiplicative depth for CT-CT operations.

Per-user access control. In multi-user deployments, different users may be authorized for different corpus subsets. We address this using additive noise cancellation within the existing CKKS scheme, requiring no additional cryptographic primitive. At ingestion, passages are classified by PII content into sensitivity classes (financial, personal, temporal, organizational). Per-class random noise $\mathbf{n}_c$ with $\|\mathbf{n}_c\| \gg \|\mathbf{e}_j\|$ is added to each restricted embedding before encryption: $\mathrm{Enc}(\mathbf{e}_j + \mathbf{n}_c)$. Each user receives correction ciphertexts: $\mathrm{Enc}(-\mathbf{n}_c)$ for authorized classes, $\mathrm{Enc}(\mathbf{r})$ (random) for unauthorized classes. These are indistinguishable under CKKS IND-CPA security. The server applies all corrections via homomorphic addition (zero multiplicative levels consumed); authorized passages are restored to $\mathrm{Enc}(\mathbf{e}_j)$ while unauthorized passages remain noise-dominated. Details and experiments in Appendix C.

Generation threat model. In the client-aided protocol (Section 8.5), the server performs matrix-vector products on CKKS ciphertexts and never observes any plaintext. The client decrypts intermediate hidden states to apply non-linear operations, then re-encrypts before sending the next projection. Since RWKV-7 weights are public, intermediate activations do not leak model information; the security goal is purely

data privacy. The server's view consists entirely of CKKS ciphertexts and Galois-rotated ciphertexts, indistinguishable from random under the RLWE assumption. GAZELLE [3] and CrypTFlow2 [18] use the same client-server split. The per-token communication pattern is fixed: exactly 96 round-trips of identically sized ciphertexts, with the same sequence of operations regardless of input content. Client-side element-wise computation ($\sim\mu$s on d=2048 vectors) is negligible relative to CKKS encryption and network transfer ($\sim$ms), limiting observable timing variation. Output length (number of tokens generated) is not hidden.

Out of scope. We do not hide access patterns, result set size, query volume, or output length. These require additional techniques (ORAM, differential privacy) orthogonal to our contribution. The client holds the secret key and decrypts similarity scores to select top-k; server-side encrypted ranking is out of scope.

3 Characterizing FHE Accuracy

Two relationships govern FHE accuracy for similarity search.

3.1 Noise Scaling

Lemma 1 (FHE Noise Scaling)**.** *For CKKS dot product of d-dimensional vectors, the noise standard deviation satisfies:*

$$\sigma_\epsilon(d) = c \cdot \sqrt{d} \tag{3}$$

where c is a constant depending on encryption parameters.

Proof sketch: The dot product $z = \sum_{i=1}^{d} x_i y_i$ sums d terms. Each term has independent noise. By variance additivity: $\text{Var}(z_\epsilon) = d \cdot \sigma^2_{\text{term}}$, giving $\sigma_\epsilon \propto \sqrt{d}$.

Empirical validation: We measure noise across dimensions 8–256 using TenSEAL CKKS (Table 1). The ratio $\sigma_\epsilon/\sqrt{d}$ is constant ($c \approx 0.0028$), confirming Lemma 1.

Table 1: CKKS noise scales as $\sqrt{d}$ ($N = 4096$, scale 2^{20}). Measured correlation ρ on random unit vectors.

Dim	σ_ϵ	$\sqrt{d}$	$c = \sigma/\sqrt{d}$	ρ (measured)
8	0.0089	2.83	0.0031	99.98%
16	0.0106	4.00	0.0026	99.94%
32	0.0150	5.66	0.0027	99.69%
64	0.0219	8.00	0.0027	98.51%
128	0.0301	11.3	0.0027	95.22%
256	0.0464	16.0	0.0029	85.45%

3.2 The Prediction Formula

Lemma 2 (FHE Similarity Correlation Bound)**.** *Let σ_z be the standard deviation of latent pairwise similarities and σ_ϵ be the CKKS noise standard deviation. The Pearson correlation between true and encrypted similarities is bounded by:*

$$\rho \leq \frac{\sigma_z}{\sqrt{\sigma_z^2 + \sigma_\epsilon^2}} \tag{4}$$

Equality holds when the CKKS error is modeled as independent, mean-zero additive noise.

This bound connects embedding geometry to FHE performance. However, the SNR bound alone is insufficient; it does not account for information lost during compression.

Corollary 1 (FHE Prediction Formula)**.** *FHE correlation is the product of compression quality and noise tolerance:*

$$\rho_{FHE} = \rho_{compression} \times \rho_{noise} \tag{5}$$

where $\rho_{compression} = corr(original\_sims, compressed\_sims)$ measures how well SVD preserves similarity structure, and $\rho_{noise} = \sigma_z/\sqrt{\sigma_z^2 + \sigma_\epsilon^2}$ is the SNR bound.

Validation: We validate the formula by comparing predicted correlation to actual CKKS correlation measured with TenSEAL across dimensions 16–64 (Appendix D). The prediction formula requires only $\rho_{\text{compression}}$ (measurable without FHE) and CKKS parameters, so embedding-dimension configurations can be evaluated without expensive FHE benchmarks (500–8,000$\times$ faster than running actual encryption).

3.3 Contrastive Failure

The prediction formula explains why contrastive compression methods fail under FHE: they produce embeddings where $\rho_{\text{compression}} \approx 0$.

Proposition 1 (InfoNCE Similarity Concentration)**.** *Let f^* minimize the InfoNCE loss*

$$\mathcal{L}_{NCE} = -\log \frac{\exp(z_i \cdot z_{j^+}/\tau)}{\sum_k \exp(z_i \cdot z_k/\tau)} \tag{6}$$

with $K \to \infty$ negatives in d dimensions. Then:

1. ***Alignment.*** *Positive pairs satisfy $f^*(x) \cdot f^*(x^+) = 1$.*
2. ***Concentration.*** *For independently drawn negative pairs, $f^*(x_i) \cdot f^*(x_j)$ has mean 0 and variance $1/d$, concentrating in $[-O(1/\sqrt{d}),\ O(1/\sqrt{d})]$ with high probability.*

Proof. Part (1): Wang & Isola [4] decompose InfoNCE into alignment and uniformity terms. The alignment term is minimized when $f^*(x) = f^*(x^+)$, giving $f^*(x) \cdot f^*(x^+) = 1$ on the unit sphere. Part (2): the uniformity term is minimized when the output distribution of f^* is uniform on S^{d-1}. For independently drawn x_i, x_j, $f^*(x_i)$ and $f^*(x_j)$ are independent uniform on S^{d-1}, so their dot product has mean 0 and variance $1/d$ with sub-Gaussian tails.

Proposition 2 (FHE Consequence)**.** *At the InfoNCE optimum, all negative similarities concentrate in a band of width*

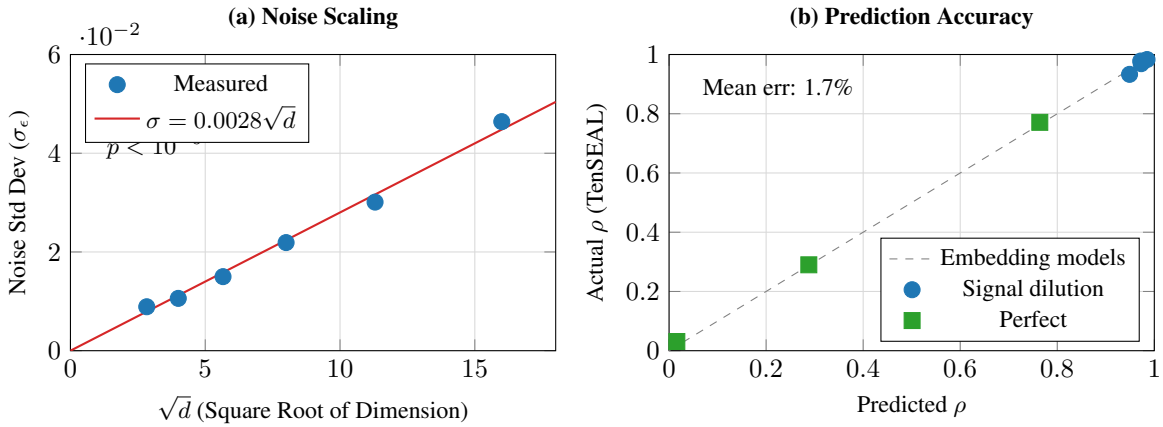

Figure 2: **FHE-Sim.** (a) Noise Scaling ($N = 4096$): CKKS noise scales as $\sigma_\epsilon = c\sqrt{d}$ with $R^2 = 0.98$ (variance additivity). (b) Prediction accuracy across embedding models and signal corruption levels (1.7% mean error over 396 configs; Appendix D).

$O(1/\sqrt{d})$ around zero. The encoded similarity variance is $\sigma_z^2 \approx 1/d$. Combined with CKKS noise $\sigma_\epsilon = c\sqrt{d}$:

$$\rho_{noise} = \frac{1}{\sqrt{1 + c^2 d^2}} \tag{7}$$

which decreases with dimension. At $d = 64$, the band is ± 0.125: a pair with original similarity 0.3 and one at 0.7 both map to values near 0, collapsing the continuous similarity range. Empirically, $\rho_{compression} \approx 0.01$ (Table 2); the concentration destroys correlation with original similarities.

High uniformity alone does not cause FHE failure. BERT embeddings have $U = 0.70$ yet achieve 97% FHE correlation because $\rho_{\text{compression}} = 0.99$. InfoNCE fails because it removes similarity structure, not because it increases uniformity.

Table 2: InfoNCE removes similarity structure ($\rho_{\text{compression}} \approx 0$).

Method	Uniformity	ρ_{comp}	ρ_{noise}	FHE
SVD (64-d)	0.64	0.99	0.98	97%
BERT (high-U)	0.70	0.99	0.98	97%
InfoNCE	0.99	0.01	0.98	1%
Random	0.99	0.01	0.98	1%

The formula $\rho_{\text{FHE}} = \rho_{\text{compression}} \times \rho_{\text{noise}}$ explains both successes and failures: InfoNCE has high ρ_{noise} but near-zero $\rho_{\text{compression}}$, yielding near-zero FHE correlation.

Failure mode detection: FHE-Sim identifies failure modes via $\rho_{\text{compression}}$:

- **Latent collapse** ($\sigma_z < 0.01$): Autoencoder failure
- **Similarity destruction** ($\rho_{\text{comp}} < 0.5$): Contrastive/random failure

Algorithm 1 FHE-Sim: Predict FHE Accuracy

1: **Input**: Embeddings X, poly_modulus_degree N, target dim d
2: Compress: $Z = \text{SVD}_d(X)$
3: Compute $\rho_{\text{comp}} = \text{corr}(X_i \cdot X_j, Z_i \cdot Z_j)$
4: Compute $\sigma_z = \text{std}(Z_i \cdot Z_j)$
5: Lookup noise constant: $c = f(N)$
6: Compute $\rho_{\text{noise}} = \sigma_z / \sqrt{\sigma_z^2 + c^2 d}$
7: **Output**: $\rho_{\text{comp}} \times \rho_{\text{noise}}$, compatibility

Table 3: FHE-Sim predictions for SVD vs. InfoNCE (2 models, d=64, N=4096).

Model	Method	Pred.	Actual	Err.
OpenAI	SVD	100%	97.8%	2.2%
OpenAI	InfoNCE	13.2%	16.1%	2.9%
BERT	SVD	100%	97.2%	2.8%
BERT	InfoNCE	16.1%	19.4%	3.3%
Mean error				**2.8%**

4 FHE-Sim

FHE-Sim predicts whether a given embedding model's similarity structure survives compression and CKKS encryption. It does not model general FHE computation; it specifically captures the interaction between embedding geometry, dimensionality reduction, and encryption noise for dot-product similarity search.

4.1 Simulation Mode

FHE-Sim can also simulate CKKS outputs by adding calibrated Gaussian noise:

$$\text{sim}_{\text{FHE}}(x, y) = \langle x, y \rangle + \mathcal{N}(0, c\sqrt{d}) \tag{8}$$

For $N = 4096$, simulated correlation matches TenSEAL CKKS within 0.5% (500–8,000× compute time reduction vs.

running actual encryption [5]). For $N = 8192$ with scale 2^{40}, CKKS noise is negligible ($\sim$1e-7) and simulation is unnecessary.

5 Complex Packing

CKKS encodes vectors into polynomial slots, where each slot holds a complex number $a + bi$. Standard vertical packing uses only the real component, leaving imaginary components unused.

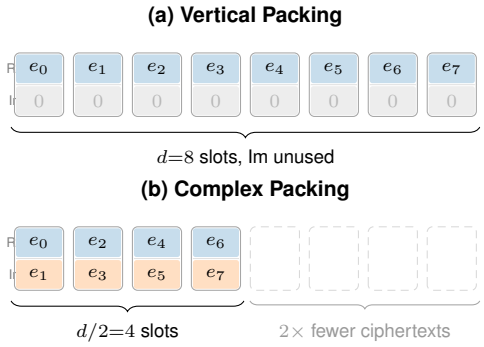

Figure 3: CKKS slot utilization for d=8. **(a)** Vertical packing stores one real value per slot, leaving imaginary components unused. **(b)** Complex packing maps consecutive pairs as $e_{2j} + ie_{2j+1}$, fully utilizing each slot and halving ciphertext count.

5.1 Packing Scheme

We pack two consecutive embedding dimensions into each slot:

$$\text{slot}_j = e_{2j} + i \cdot e_{2j+1} \tag{9}$$

For a d-dimensional embedding, this requires $d/2$ slots instead of d.

Queries are packed as conjugates:

$$\text{slot}_j^{(q)} = q_{2j} - i \cdot q_{2j+1} \tag{10}$$

5.2 Inner Product Recovery

The ciphertext-plaintext product yields:

$$(e_r + i \cdot e_i)(q_r - i \cdot q_i) = (e_r q_r + e_i q_i) \tag{11}$$

$$+ i(e_i q_r - e_r q_i) \tag{12}$$

The real part equals $e_r q_r + e_i q_i$, the contribution of two dimensions to the inner product. Summing real parts across slots gives the full inner product.

Complex packing is mathematically identical to vertical packing but halves ciphertext count. Compared to standard approaches using 768-dimensional embeddings (e.g., BERT-base) with vertical packing, our 64d complex-packed representation uses $768/32 = 24\times$ fewer ciphertexts.

6 Hyperbolic Embeddings

Euclidean space may not be optimal for embedding hierarchical or tree-structured data. Hyperbolic geometry provides exponentially more volume at a given radius, better matching the growth rate of hierarchical structures [8].

6.1 The Lorentz Model

We use the Lorentz (hyperboloid) model $\mathcal{L}^n = \{x \in \mathbb{R}^{n+1} : \langle x, x\rangle_{\mathcal{L}} = -1, x_0 > 0\}$ where the Lorentz inner product is:

$$\langle x, y\rangle_{\mathcal{L}} = -x_0 y_0 + \sum_{i=1}^{n} x_i y_i \tag{13}$$

For FHE, this reduces to a standard dot product with a sign flip on the first component; no transcendental functions are needed.

6.2 FHE-Compatible Similarity

For points on the hyperboloid, the hyperbolic distance is $d_{\mathcal{L}}(x, y) = \text{arcosh}(-\langle x, y\rangle_{\mathcal{L}})$. Since arcosh is monotonic, ranking by $-\langle x, y\rangle_{\mathcal{L}}$ preserves the distance ordering.

Under CKKS with complex packing, we encode:

$$d_j = d_{2j} + i \cdot d_{2j+1} \tag{14}$$

$$q_j = \begin{cases} -q_0 + i \cdot q_1 & j = 0 \text{ (time component negated)} \\ q_{2j} - i \cdot q_{2j+1} & j > 0 \text{ (conjugate)} \end{cases} \tag{15}$$

The real part of $\sum_j q_j \cdot d_j$ yields $-q_0 d_0 + \sum_{i>0} q_i d_i = \langle q, d\rangle_{\mathcal{L}}$.

6.3 Euclidean to Hyperbolic Projection

Given Euclidean embeddings $e \in \mathbb{R}^d$, we project to the hyperboloid via:

$$x = \left(\sqrt{1 + \|We\|^2},\ We\right) \tag{16}$$

where $W \in \mathbb{R}^{n \times d}$ is learned. The time component $x_0 = \sqrt{1 + \|x_{1:n}\|^2}$ ensures $\langle x, x\rangle_{\mathcal{L}} = -1$.

On MS MARCO, Lorentz hyperbolic embeddings achieve 90.5% R@10 vs 88.0% Euclidean (+2.5%), with identical FHE operations. Beyond accuracy, the Lorentz model enables complex packing (Eq. 9–10): pairing coordinates as real and imaginary parts halves slot usage per document, doubling SIMD throughput (e.g. 124 vs 62 documents per ciphertext at N=8192). This 2× packing is used throughout retrieval and generation (Section 8.5).

7 Experiments

7.1 Setup

Retrieval benchmark. MS MARCO passage retrieval: 1,214 queries, 10,001 documents. Strict train/test split: queries 0–799 for training projections, 800+ for evaluation. All reported metrics are on the test set.

Embeddings. Qwen3-Embedding-0.6B (1024d, MRL-trained), OpenAI text-embedding-3-large (1536d), all-MiniLM-L6-v2 (384d).

FHE implementations. CPU: TenSEAL/Pyfhel with $N = 4096$. GPU: PhantomFHE [9] on NVIDIA RTX 3090 and A100 80GB (N up to 32768). We implemented CKKS bootstrapping for PhantomFHE (CoeffToSlot/EvalMod/SlotToCoeff with direct RNS scalar multiplication) and extended its Python bindings with a `cuDoubleComplex` type caster for complex packing.

Baselines. SVD (no centering), random projection, learned linear projections with InfoNCE and ranking distillation losses.

7.2 Similarity Preservation

Before evaluating retrieval, we verify that compression preserves similarity structure. Using OpenAI `text-embedding-3-large` (1536d), we measure Pearson correlation between original and FHE-computed similarities.

Table 4: Similarity correlation (N=4096). Higher is better.

Dim	RP	SVD	Learned	MLP
32	82.3%	94.7%	95.7%	94.2%
64	86.8%	96.8%	96.9%	96.2%
128	91.7%	97.4%	97.2%	—

SVD without centering matches learned methods. Standard PCA implementations center data before projection; for unit-normalized embeddings, this can degrade similarity structure (effect varies by embedding characteristics):

Table 5: Effect of centering on similarity correlation (OpenAI, 64d). The gap ranges from 7–24% depending on embedding characteristics.

Method	Correlation
`sklearn.PCA` (centered)	73.2%
SVD / TruncatedSVD (no centering)	**97.1%**

7.3 Recall@k

We evaluate retrieval quality using Recall@k on the MS MARCO test set.

Qwen3-Embedding is trained with Matryoshka Representation Learning (MRL) [7], so truncating to the first 64 of 1024 dimensions yields a valid embedding without any learned projection. Moving from generic embeddings (65.2%) to MRL truncation (92.3%) yields the largest gain. Ranking distillation (Appendix A) contributes an additional 2.7 percentage points.

Table 6: Recall@k comparison (MS MARCO test set, 64d).

Method	R@10	R@100
MS MARCO 384d + SVD 64d	65.2%	89.1%
MS MARCO 384d + learned 64d	65.2%	—
Qwen3 MRL truncate 64d	92.3%	99.5%
Qwen3 distilled 64d	94.8%	99.7%
Qwen3 combined 64d	**95.0%**	**99.7%**
Qwen3 MRL truncate 128d	96.2%	99.8%
Qwen3 full 1024d	98.2%	100%

7.4 CT-CT: Encrypted Similarity Joins

In CT-CT mode, both query and document embeddings are encrypted. The server computes similarities via ciphertext-ciphertext multiplication without observing either party's data.

SIMD Batching. Complex packing yields 33 slots per 64d Lorentz embedding. With $N = 8192$ providing 4096 slots, we pack $\lfloor 4096/33 \rfloor = 124$ documents per ciphertext. A single ciphertext-ciphertext multiplication computes 124 similarities in parallel.

Table 7: CT-PT vs CT-CT performance with GPU acceleration and SIMD batching.

Documents	CT-PT	CT-CT	Per-doc	Ranking
1,000	17ms	16ms	16μs	10/10
10,000	143ms	131ms	14μs	10/10
50,000	835ms	630ms	13μs	10/10

CT-CT achieves 13–16μs per document at 64 dimensions, giving sub-second retrieval over 50,000 documents. CT-CT is faster than CT-PT (0.76–0.94×) because pre-encrypted documents avoid repeated plaintext encoding overhead during query processing. The "Ranking" column verifies FHE correctness: encrypted rankings match plaintext rankings exactly.

7.5 Latency and Throughput

Table 8: PT-CT latency comparison (CPU, TenSEAL). Compression allows a smaller polynomial modulus.

Configuration	Latency	Size	QPS
1536-d (N=8192)	31.4ms	326KB	32
1536-d (N=4096)	8.6ms	79KB	116
64-d (N=4096)	4.0ms	79KB	252

Dimensionality reduction provides 7.9× speedup over production parameters on CPU. For CT-CT with GPU acceleration, see Table 7.

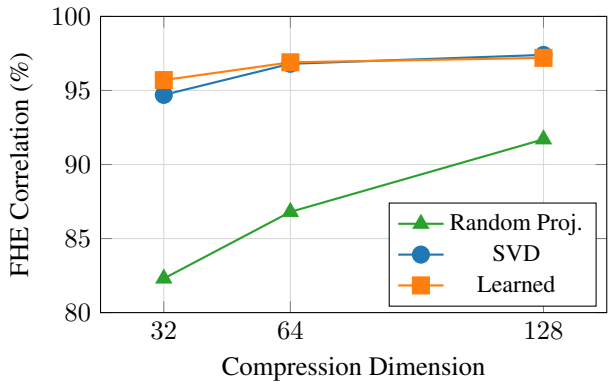

Figure 4: Similarity correlation vs dimension. SVD preserves 97% correlation at 64d. For retrieval metrics (R@10), see Table 6.

8 State Space Models for FHE Generation

While retrieval under FHE is addressed by the techniques above, *generation* under FHE has remained impractical. Transformer architectures require softmax attention: $\text{softmax}(QK^\top/\sqrt{d})V$, where $\text{softmax}(x)_i = e^{x_i}/\sum_j e^{x_j}$. Both exponentials and divisions are prohibitively expensive under CKKS, and the $O(n^2)$ attention complexity compounds the cost.

State space models (SSMs) offer a fundamentally different computational structure. Linear recurrent architectures like RWKV [10] and Mamba [11] use only:

1. Matrix-vector multiplications
2. Element-wise products
3. Gating activations (sigmoid, exp)

While certain gating activations (sigmoid, exp) require polynomial approximation, the dominant feedforward nonlinearity in RWKV (squared activation) can be computed exactly via ciphertext–ciphertext multiplication.

Polynomial approximation of SSM nonlinearities. We verify that all RWKV-7 gating functions (sigmoid, tanh, exponential decay $e^{-e^{-0.5}\cdot\sigma(w)}$) are polynomial-friendly. Table 9 shows Chebyshev polynomial approximation across 15 diverse prompts and 20 tokens each, replacing all nonlinearities simultaneously during full 1.5B model inference (d=2048, 24 blocks). At degree 19, every generated token matches exact arithmetic.

Table 10 evaluates sigmoid polynomials on CKKS ciphertexts (A100, N=32768, L_0=20, 40-bit primes), using binary powering for depth-efficient power basis construction. Degree 11–13 consumes only 5 multiplicative levels with sub-percent error, comparable to a single BSGS matmul call.

By contrast, transformer softmax requires normalization by $\sum_j e^{x_j}$, which involves $1/\sqrt{\cdot}$ over encrypted values. Profiling RWKV-7's normalization layers (LayerNorm, GroupNorm) reveals input variance spanning 5 orders of magnitude ($[6\times 10^{-4},\ 2\times 10^{4}]$), making polynomial approximation of $1/\sqrt{\cdot}$ infeasible at any practical degree. SSM gating functions operate on bounded, well-conditioned inputs ($[-8, 8]$ for sigmoid/tanh); transformer normalization does not.

Table 9: Polynomial approximation of RWKV-7 nonlinearities (1.5B model, 24 blocks, 15 prompts × 20 tokens). Chebyshev polynomials replace sigmoid, tanh, and exponential decay simultaneously.

Degree	Prompts	Tokens	Match Rate
11	11/15	268/300	89.3%
13	12/15	271/300	90.3%
15	12/15	271/300	90.3%
19	**15/15**	**300/300**	**100%**

Table 10: Sigmoid polynomial evaluation under CKKS (A100, 100 test points on $[-6, 6]$). Levels = multiplicative depth consumed.

Degree	Levels	Time	Max Err	Corr.
7	4	35 ms	2.2×10^{-2}	0.9995
11	5	53 ms	4.9×10^{-3}	0.99998
13	**5**	**62 ms**	$\mathbf{2.3\times10^{-3}}$	**0.99999**

8.1 Fully Encrypted Inference

We run RWKV-7 (1.5B) token prediction under CKKS, using trained model weights. No intermediate decryptions; logits are decrypted only at the output. Unless otherwise noted, all configurations in this section achieve exact functional agreement with plaintext inference (correlation 1.0, identical predicted tokens).

The single-block pipeline consists of: embedding lookup (plaintext), encryption of the hidden state, FFN key projection (CT-PT), ciphertext–ciphertext squaring activation (CT-CT), FFN value projection (CT-PT), head projection (CT-PT), and decryption of final logits followed by argmax. For numerical stability under CKKS, we apply column-wise variance normalization to weights and scale embeddings by a constant factor.

The squaring operation corresponds to the dominant nonlinear component of RWKV's squared-ReLU feedforward block. We evaluate configurations up to 1024 hidden dimensions and 2048 intermediate dimensions using CKKS with poly modulus degree 32768 and scale 2^{40}.

Table 11: Fully encrypted RWKV inference with x^2 activation (RTX 3090).

Hidden×FFN×Vocab	Activation	Time
64×128×32	x^2	7.6s
128×256×64	x^2	27.7s
256×512×64	x^2	96.3s
512×1024×64	x^2	355.8s
1024×2048×64	x^2	1401.8s

Runtimes scale with the number of encrypted dot products; the nonlinear activation is not the dominant cost.

8.2 Multi-Layer FHE Inference

We extend to multi-layer inference by chaining N feedforward blocks under a single encryption, with no intermediate decryption. Each block consumes 3 multiplicative levels (key projection, squaring, value projection), plus 1 for the final head projection, giving a total depth of $3N + 1$.

Block 0 operates on a packed ciphertext (full hidden vector in one ciphertext) using `ct_pt_dot` with rotate-and-sum for the key projection. The output is a *list* of ciphertexts (one scalar per FFN dimension). Subsequent blocks operate on this list representation, using `ct_pt_weighted_sum` for all projections. This avoids costly repacking while preserving the exact same FHE primitives.

The modulus chain $[60, 40 \times d, 60]$ with one special prime provides d multiplicative levels. At $N = 32768$, security constraints allow up to $d = 19$ levels (880 bits total modulus), supporting $\lfloor (19-1)/3 \rfloor = 6$ chained blocks. We evaluate configurations from 2 to 6 blocks using weights from consecutive RWKV-7 layers:

Table 12: Multi-layer FHE RWKV inference (RTX 3090, no intermediate decryption). Each block: key projection, x^2 activation, value projection. † magnitude-controlled normalization.

Config	Blocks	Depth	Time	Mag-Ctrl
64×128×32	2	7/9	28.4s	
128×256×64	2	7/9	142.0s	
64×128×32	3	10/11	75.1s	
128×256×64	3	10/11	186.9s	
64×128×32	4	13/14	63.2s	†
64×128×32	5	16/17	83.1s	†
64×128×32	6	19/20	104.6s	†
128×256×64	4	13/14	321.6s	†

Magnitude-controlled normalization. Without normalization, cascaded squaring causes exponential magnitude growth: $|h|_{\max} \approx 10^6$ after 2 blocks, 10^{14} after 3, and 10^{32} after 4, causing CKKS precision loss (correlation ≈ 0.04 at 4 blocks). We solve this by absorbing per-block scaling factors into W_{val}: after computing plaintext magnitudes on one calibration input, we set $W_{\text{val}}^{(b)} \leftarrow W_{\text{val}}^{(b)} \cdot (\alpha / |h^{(b)}|_\infty)$ for a target magnitude $\alpha = 10$. Since `argmax` is scale-invariant, the predicted token is identical. This adds zero extra multiplicative depth: the scaling is folded into existing plaintext multiplications. With magnitude control, 4–6 blocks all match plaintext, extending the practical depth from 10 to 19 levels within the same $N = 32768$ parameter set.

Residual connections. Ciphertext-level skip connections for blocks 1+ are achieved via `mod_switch_to_next` (level alignment, 0 extra depth) and `set_scale` (scale matching), followed by ciphertext addition. Both 2- and 3-block configurations with residuals match plaintext at zero additional multiplicative cost.

Autoregressive generation. Multi-token generation: each token step encrypts the embedding, runs multi-layer FHE inference, decrypts logits, and applies argmax. The client (who holds the secret key) selects each token; the server never observes any plaintext token. With 2-block inference, 5 consecutive tokens match plaintext (191.0s total, RTX 3090); with 3 blocks, 3 tokens match (134.0s total).

Time-mixing under FHE. RWKV's time-mixing path computes $\sigma(r) \odot (k \cdot v)$ projected through W_o, where σ is the sigmoid gate. We approximate $\sigma(x) \approx 0.25x + 0.5$ (linear sigmoid), consuming 0 extra multiplicative levels since it decomposes into a plaintext scaling and addition. Level alignment requires care: r and k are both at 1 rescale after their respective `ct_pt_dot` projections, so $r \cdot k$ (CT-CT) produces a result at 2 rescales. We then align v via `mod_switch_to_next` (0 extra depth) before computing $(r \cdot k) \cdot v$ at 3 rescales. One block of time-mixing + FFN uses 7 multiplicative levels (4 for time-mix, 3 for FFN, 24.5s on RTX 3090). Multi-block time-mixing requires magnitude control for the combined path.

8.3 Scaling to Full Vocabulary

The experiments above use reduced vocabulary sizes (32–64 tokens) because the head projection $h \mapsto W_{\text{head}} h$ requires one encrypted weighted sum per vocabulary token. At the full RWKV-7 vocabulary of 65,536 tokens, computing 65,536 encrypted weighted sums is prohibitively expensive.

Client-side head completion. The head projection need not be computed under FHE. Since the client already decrypts all logits to apply argmax, the server can simply return the encrypted hidden state h. The client decrypts h (d_{embed} values), applies $W_{\text{head}} \in \mathbb{R}^{d_{\text{embed}} \times V}$ in plaintext, and takes argmax. This introduces no approximation, supports any vocabulary size, and saves one multiplicative level (depth $3N$ instead of $3N + 1$ for N blocks). The security model is unchanged: the server never observes any plaintext, and the client already receives the logit vector in all protocols.

Batched FHE computation. At full model dimensions ($d_{\text{embed}} = 2048$, $d_{\text{ffn}} = 8192$), holding all intermediate ciphertexts simultaneously exceeds GPU memory. We process FFN dimensions in batches of 1024, fusing key projection and squaring for each dimension (avoiding storing both ct_k and ct_{k^2} simultaneously) and accumulating partial value projections across batches via ciphertext addition. Peak memory drops from $O(2 \times d_{\text{ffn}})$ to $O(b + d_{\text{embed}})$ ciphertexts.

Table 13: Full-vocabulary FHE inference (V=65,536, RTX 3090) with client-side head. † magnitude-controlled.

Hidden×FFN×Vocab	Blocks	Depth	Time
64×128×65536	1	3/5	5.9s
128×2048×65536	1	3/5	177.7s
1024×2048×65536	1	3/5	1381.0s
1024×2048×65536†	2	6/8	4096.9s
2048×8192×65536	1	3/5	10863.4s

Client-side head completion at $V = 65{,}536$ takes <1s in

plaintext, avoiding 65,536 encrypted weighted sums. The 1024×2048 configuration completes one token in 23 minutes on a single consumer GPU (RTX 3090); since FFN dimensions are processed independently, the workload parallelizes linearly.

8.4 Encrypted RAG Pipeline

We combine encrypted retrieval (Section 7) with encrypted generation into a single private RAG pipeline.

Protocol. The pipeline uses two CKKS contexts optimized for each phase:

1. **Encrypted retrieval** ($N = 8192$, complex packing): The client encrypts the query embedding; the server computes batched similarity scores against all documents via SIMD packing (124 docs per ciphertext). In CT-PT mode, documents remain as plaintexts; in CT-CT mode, documents are pre-encrypted, providing full bilateral encryption. The client decrypts scores and selects the top document.

2. **Encrypted generation**: At small dimensions (top rows of Table 14), the server runs fully encrypted multi-block inference with no intermediate decryptions. At full scale (bottom rows), the client-aided BSGS protocol (Section 8.5) is used, where the server performs matrix-vector products on encrypted data and the client handles non-linear operations between projections.

The two phases share no cryptographic state; the only bridge is the client's plaintext decision of which document was retrieved. In both cases, the server never observes any plaintext query, document, or generated token.

Table 14: Encrypted RAG pipeline. Top: fully encrypted at small dimensions. Bottom: client-aided BSGS at full scale (A100). All generated tokens match plaintext (correlation 1.0).

Config	Blk	Docs	Mode	Ret.	Gen.
64×128×32	2	100	CT-PT	48ms	82.1s
64×128×32	3	200	CT-PT	50ms	145.7s
128×256×64	2	100	CT-PT	76ms	336.6s
64×128×32	2	100	CT-CT	69ms	96.6s
2048×8192×65k	24	100	CT-PT	1.0s	429s/tok
2048×8192×65k	24	100	CT-CT	1.0s	429s/tok
1024×4096×65k†	24	100	CT-PT	1.0s	23s/tok

†0.4B model, N=8192, CPU-offloaded diagonals. 50 queries (30 SQuAD + 20 MS-MARCO), all 50/50 tokens match plaintext.

Retrieval latency is dominated by a single batched ciphertext multiplication (<100 ms at small dimensions; <4 s at full scale). In CT-CT mode both query and documents are encrypted, providing bilateral privacy.

The full-scale rows use client-aided BSGS (Section 8.5). On 100 SQuAD passages with 3 queries at d=2048, CT-PT and CT-CT both achieve 3/3 gold recall at rank 1, and every generated token matches plaintext RWKV-7 (correlation 1.0). Token-matching means QA metrics under FHE equal the plaintext model's. At d=1024 (0.4B) with CPU-offloaded pre-encoded diagonals, we scale to 50 queries across both datasets: 30/30 SQuAD and 20/20 MS-MARCO queries produce FHE tokens identical to plaintext (correlation 1.0), with 23 s/token on the A100. Retrieval recall (16/30 SQuAD, 19/20 MS-MARCO at R@1) reflects the embedding model's capacity at 100 passages rather than the encryption pipeline. Neither RWKV-7 1.5B nor comparably sized models (e.g. Qwen2.5-1.5B [19], MMLU 45–46) are tuned for extractive QA; these scores reflect the base model's capacity at this scale.

8.5 Client-Aided Generation with BSGS Acceleration

The fully encrypted approach (Tables 11–13) computes all operations under a single encryption without intermediate decryption, but is limited to shallow depth by the CKKS modulus chain. To scale to the full 24-block RWKV-7 (1.5B parameters, $d = 2048$, $d_{\text{ffn}} = 8192$), we adopt a *client-aided protocol* combined with the baby-step giant-step (BSGS) diagonal method for matrix-vector products.

Protocol. Each RWKV block is partitioned between server and client:

1. **Server (FHE):** computes the six large matrix-vector products (r, k, v, o projections at $d \times d$; FFN key at $d \times d_{\text{ffn}}$; FFN value at $d_{\text{ffn}} \times d$) using CKKS-encrypted inputs.

2. **Client (plaintext):** decrypts server outputs and performs all non-linear operations: LayerNorm, GroupNorm, sigmoid gating, WKV state update, and ReLU^2 activation.

The client re-encrypts each intermediate result before sending it back for the next server projection. Per block, this requires 4 client-server round-trips (r,k,v jointly, then o, then FFN key, then FFN value), giving 96 round-trips per token across 24 blocks. The server never observes any plaintext hidden state.

Computation split. The server holds the six projection matrices per block (∼2.7 GB for 24 blocks in float16) and performs all $O(d^2)$ and $O(d \cdot d_{\text{ffn}})$ matrix-vector products under FHE. The client holds the embedding table, output head, and per-block normalization/gating vectors (∼550 MB), performing only $O(d)$ element-wise operations (LayerNorm, sigmoid, WKV update, ReLU^2) that take <0.01s per block. The client offloads the expensive linear algebra to a GPU server while keeping all data encrypted, following the outsourced computation model of GAZELLE [3]. Since the model weights are public (RWKV-7 is Apache-licensed), the security goal is data privacy, not model privacy: the server must not learn the client's query, retrieved documents, or generated text.

BSGS diagonal method. For $y = Wx$ where $W \in \mathbb{R}^{d \times d}$, the standard approach computes each output element independently via a rotate-and-sum dot product, requiring $d \cdot \lceil \log_2 d \rceil$ rotations per projection. The diagonal method [13] decomposes W into d diagonals $\delta_k[j] = W[j, (j{+}k) \bmod d]$ and computes $y = \sum_{k=0}^{d-1} \text{rotate}(x, k) \odot \delta_k$, encoding all d output elements in a single ciphertext via SIMD packing. Baby-step giant-step factoring writes $k = gG + b$ with $G = \lceil \sqrt{d} \rceil$, $B = \lceil d/G \rceil$, reducing the rotation count to $G{+}B{-}2$ (identity rotations require

no key). At $d = 2048$: $G = 46$, $B = 45$, giving $G+B-2 = 89$ rotations per projection versus $2048 \times 11 = 22{,}528$ in the naive approach, a 253× reduction.

Complex packing. For the FFN key projection ($d \to d_{\text{ffn}} = 4d$), we split the output into 4 chunks of d elements each. Since the input is real-valued, we pack two weight chunks as real and imaginary parts of complex-valued diagonals, computing two output chunks per BSGS call, reducing the cost from 4 to 2 calls. For the FFN value projection ($d_{\text{ffn}} \to d$), the 4 input chunks are paired using the conjugate trick: we encrypt $\text{Enc}(x_0 + ix_1)$ and encode diagonals as $(\delta_0 - i\delta_1)$ so that $\text{Re}(\text{BSGS}(\cdot)) = M_0x_0 + M_1x_1$, again reducing 4 real BSGS calls to 2 complex ones. Both directions share baby-step rotations within each pair, and partial results accumulate in plaintext.

Each BSGS call consumes exactly 1 multiplicative level (one `rescale_to_next`), independent of d, versus d rescales in the naive per-element approach. This eliminates the depth bottleneck that previously limited inference to a single block at full model dimensions.

Table 15: Client-aided BSGS generation across model sizes, CKKS parameters, and GPUs. All configurations produce identical tokens under FHE and plaintext (3 tokens, 24 blocks). The 1.5B generates "*is Paris.*" on the RTX 3090; both models generate "*Paris*" on the A100. †CPU-offloaded plaintexts with FFN value complex packing (8 BSGS calls/block).

Model (d)	GPU	N	Server/blk	Client/blk	Per token
1.5B (2048)	RTX 3090	32768	20.6s	<0.01s	494s (8.2 min)
1.5B (2048)	A100 80GB	16384	14.1s	<0.01s	338s (5.6 min)
1.5B (2048)	A100 80GB	8192	8.9s	<0.01s	213s (3.6 min)
1.5B (2048)	A100 80GB†	8192	3.1s	<0.01s	79s (1.3 min)
0.4B (1024)	A100 80GB	16384	6.6s	<0.01s	160s (2.7 min)
0.4B (1024)	A100 80GB	8192	4.0s	<0.01s	96s (1.6 min)
0.4B (1024)	A100 80GB†	8192	0.8s	<0.01s	19s

Client computation is negligible (<0.1% of total time). Each of the 24 blocks requires 8 BSGS calls (4 for attention projections, 2 complex-packed for FFN key, 2 complex-packed for FFN value). Pre-encoding all diagonal plaintexts on the CPU and streaming them to the GPU via a persistent staging buffer eliminates per-call device allocation overhead, reducing per-block server time from 8.9s to 3.1s on the A100 at N=8,192. The per-token latency scales linearly with block count and is independent of vocabulary size (client-side head completion).

Parameter tuning. In the client-aided protocol, each BSGS matmul consumes exactly one multiplicative level; the client decrypts and re-encrypts between projections, resetting the level budget. This means only L_0=3 working levels are needed (down from 5), and the polynomial degree can be reduced from N=16,384 to N=8,192 while remaining within the HE Standard's 128-bit security bound ($\log_2 Q = 216 \leq 218$ for N=8,192). The smaller polynomial degree halves the NTT size, reducing per-block latency from 14.1s to 8.9s for the 1.5B model and from 6.6s to 4.0s for the 0.4B model. Combining both optimizations (smaller model and smaller N) yields a 3.5× overall speedup.

Compared to the fully encrypted single-block approach (Table 13), the full 24-block model under the client-aided protocol completes one token (19–494s) in less time than a single naive block at the same dimensions (10,863s). The per-block speedup arises from BSGS rotation reduction (89 vs ∼22,500) and client-side non-linearities. The cost is 96 client-server round-trips per token, each transmitting one ciphertext (∼4 MB at N=32768, ∼1 MB at N=8192).

Full-scale RAG. Combining BSGS-accelerated generation with the encrypted retrieval pipeline (Section 7), we run retrieval + generation on SQuAD v2 and MS-MARCO. At d=2048 (100 passages, 3 queries, 3 tokens each), both CT-PT and CT-CT retrieval achieve 3/3 gold recall at rank 1 (429 s/token). At d=1024 with CPU-offloaded pre-encoded diagonals (23 s/token), we scale to 50 queries: 30 SQuAD and 20 MS-MARCO, each generating 3 tokens. All 50/50 queries produce FHE tokens identical to plaintext (correlation 1.0). Retrieval R@1 is 16/30 on SQuAD and 19/20 on MS-MARCO (Table 14).

Token count. Since FHE tokens match plaintext exactly (correlation 1.0), the useful token count is limited by the model, not the encryption. At d=2048 (full 1.5B, 24 blocks), greedy decoding on 10 diverse prompts produces 30 coherent tokens each with no token-level degeneration; sentence-level repetition (common in greedy LM decoding) is the only quality-limiting factor.

Retrieval mode ablation. To verify that encryption introduces no ranking errors, we run retrieval-only benchmarks with 100 passages and 10 queries in three modes: plaintext (no encryption), CT-PT (encrypted query, plaintext documents), and CT-CT (both encrypted). Table 16 confirms that all three modes produce identical rankings on both datasets.

Table 16: Retrieval mode ablation (100 passages, 10 queries). All three modes produce identical rankings. Latency is per-query with GPU-accelerated SIMD batching.

Dataset	Mode	R@1	R@5	R@10	ms/q
SQuAD	Plaintext	80%	90%	100%	1.3
SQuAD	CT-PT	80%	90%	100%	1.8
SQuAD	CT-CT	80%	90%	100%	1.4
MS-MARCO	Plaintext	90%	90%	100%	0.2
MS-MARCO	CT-PT	90%	90%	100%	1.6
MS-MARCO	CT-CT	90%	90%	100%	1.4

Retrieval adds <4 s per query. Generation latency (429–443 s/token at N=16,384) is dominated by per-block BSGS computation (Table 15); retrieval overhead is <1%.

Plaintext prefill. In the RAG setting, the client already possesses the retrieved passage (having decrypted the retrieval result). Rather than feeding only a short seed token into FHE generation, the client can construct a full prompt ("Context: {passage}\nQuestion: {query}\nAnswer:") and process it entirely through the plaintext RWKV forward pass, building up the recurrent WKV state across all 24 blocks. Only the answer generation phase then runs under FHE. Since RWKV's state is a fixed-size $d_{\text{head}} \times d_{\text{head}}$ matrix per head per block (independent of sequence length), the full context is compressed into

the state; the server receives an encrypted state that encodes the complete passage without knowing its content. Prefilling a 200-token prompt takes <2 s in plaintext on CPU, after which the model generates short answers ("Paris" for a factual question) in as few as 1–3 FHE tokens (19–494 s per token depending on GPU and model size). This amortizes context processing to <10% of total latency even at the fastest configuration.

8.6 Fully Encrypted BSGS Inference with Bootstrap

The client-aided protocol (Section 8.5) achieves exact inference at full model scale but requires 96 round-trips per token. BSGS also supports fully encrypted multi-block inference (no intermediate decryptions) at $d = 2048$, $d_{\text{ffn}} = 8192$.

Pipeline. Each fully encrypted FFN block consumes exactly 3 multiplicative levels: (1) BSGS key projection (d by d_{ffn}) via 4 chunk-wise BSGS calls sharing baby-step rotations (1 level); (2) ciphertext–ciphertext squaring activation (1 level); (3) BSGS value projection (d_{ffn} by d) via 4 chunk-wise BSGS calls with partial accumulation under FHE (1 level). Magnitude control is pre-folded into W_{val} at zero extra depth. Residual connections use level-aligned ciphertext addition.

Without bootstrap. Using a deep modulus chain ($L_0 = 60$, 59-bit primes, $N = 32768$, $P = 3$), we obtain 59 usable multiplicative levels, supporting $\lfloor 59/3 \rfloor = 19$ fully encrypted FFN blocks with no bootstraps. On an A100 GPU:

Table 17: Fully encrypted BSGS inference at d=2048, d_{ffn}=8192 on A100 80GB. No intermediate decryption. "BT" = number of CKKS bootstraps performed.

N	L_0	Blocks	BT	Corr.	Max Err	Time/blk
32768	60	19/19	0	**1.000000**	9.3×10^{-9}	70s
16384	36	24/24	4	0.999989	1.0×10^{-1}	40s

Without bootstrap, all 19 blocks achieve correlation 1.0 with plaintext, with maximum absolute error 9.3×10^{-9}, comparable to CKKS encoding precision. Error grows linearly from 8.8×10^{-11} (block 0) to 9.3×10^{-9} (block 18), accumulating only from CKKS rounding noise across $19 \times 3 = 57$ rescale operations.

With CKKS bootstrap. To reach the full 24-block depth, we employ CKKS bootstrapping [14] (level budget $[2, 2]$, depth 20) to refresh the modulus chain. At $N = 16384$, $L_0 = 36$, $P = 3$: the first 11 blocks run without bootstrap, then 4 bootstraps refresh the chain for blocks 11–23. Each bootstrap introduces $\sim$0.025 error (at the ciphertext magnitude $\sim$4), accumulating linearly across 4 bootstraps. The final correlation after 24 blocks is 0.99999, with total computation time $24 \times 40\text{s} + 4 \times 0.7\text{s} = 963\text{s}$ (16.1 min).

Divisibility requirement. PhantomFHE requires $L_0 \equiv 0 \pmod{P}$ (the data prime count must be divisible by the special modulus count). Non-divisible values cause silent memory errors during modular switching. Bootstrap keygen memory scales with $L_0 \times N \times |\text{Galois elements}|$; the maximum L_0 for bootstrap on A100 (80GB) is 45 at $N = 32768$ and 36 at $N = 16384$ with 161 Galois elements.

8.7 Contrastive SSM Embeddings

For retrieval, contrastive training improves SSM embedding quality [12]. On MS-MARCO (1000 queries, 3909 documents, 80/20 train/test split):

Table 18: Embedding quality at 64 dimensions after SVD projection (subset evaluation: 1000 queries, 3909 docs).

Model	R@10	R@100
Contrastive SSM (0.4B)	**99.3%**	100%
Qwen3-Embedding (MRL truncate)	90.8%	99.5%

Contrastive-trained SSM embeddings maintain 99.3% R@10 under 16× dimensionality reduction, compared to 90.8% for MRL-truncated transformer embeddings at the same dimension.

9 Related Work

FHE for Machine Learning. CryptoNets [2], GAZELLE [3], and subsequent systems (NEXUS, BOLT, EncryptedLLM) perform encrypted neural inference on classification or single forward passes. FHE-SPEAR extends this to multi-token autoregressive generation, where each decoded token feeds back as input. Our Lemma 1 characterizes noise accumulation in high-dimensional dot products. Halevi and Shoup [13] introduced the BSGS diagonal method for FHE matrix-vector multiplication, reducing rotations from $O(d)$ to $O(\sqrt{d})$; we apply this at d=2048 with complex packing.

FHE-Based RAG. SecureRAG [15] provides FHE-encrypted retrieval with document-level access control via lattice-based KP-ABE (PALISADE-abe). FRAG [16] addresses encrypted approximate nearest neighbor search across federated vector databases. HyFedRAG [17] uses TenSEAL for embedding privacy in a federated setting but runs generation on plaintext LLMs. These systems focus on the retrieval side of RAG; FHE-SPEAR complements this with encrypted generation via client-aided RWKV-7 inference under CKKS with BSGS acceleration. For access control, SecureRAG uses KP-ABE as a separate cryptographic layer with document-level granularity, where the server observes which ABE keys a user holds. FHE-SPEAR uses additive noise cancellation within the existing CKKS scheme (Appendix C), operating at passage-level granularity with no additional library. Because real and dummy correction ciphertexts are indistinguishable under CKKS IND-CPA, the server cannot determine a user's authorization level. Both approaches consume zero multiplicative levels.

CKKS Noise Analysis. Prior work analyzes CKKS noise growth through multiplicative depth [1]. Cheon et al. [14] introduced CKKS bootstrapping to refresh the modulus chain, enabling arbitrary-depth computation at the cost of approximation error per refresh. Noise also accumulates with vector

dimension in dot products, a distinct phenomenon relevant to similarity search.

Hyperbolic Embeddings. Nickel & Kiela [8] introduced Poincaré embeddings for hierarchical data. We use the Lorentz model, whose bilinear form is directly computable under FHE.

Contrastive Learning. Wang & Isola [4] proved that InfoNCE is minimized by uniform representations on the hypersphere. Proposition 1 shows this concentrates all negative similarities at zero, destroying similarity values required for FHE (Proposition 2).

Rate-Distortion Theory. Classical bounds characterize compression limits [6]. Under FHE, noise accumulation creates an additional constraint not captured by standard theory.

State Space Models. RWKV [10] and Mamba [11] avoid softmax attention, eliminating the most expensive polynomial approximation in encrypted transformer inference. Their fixed-size recurrent state ($O(d^2)$ vs. a growing $O(nd)$ KV cache) gives constant per-token memory under FHE.

10 Conclusion

We presented techniques for private retrieval-augmented generation under CKKS homomorphic encryption.

Complex packing encodes two dimensions per slot, halving ciphertext count. Contrastive-trained SSM embeddings maintain 99.3% R@10 at 64 dimensions under SVD projection, compared to 90.8% for MRL-truncated Qwen3-Embedding on the same subset.

For generation, state space models avoid softmax entirely, computing the dominant nonlinearity (x^2) via a single ciphertext-ciphertext multiplication with no polynomial approximation. RWKV's fixed-size $O(d^2)$ state (vs. a growing $O(nd)$ KV cache) gives constant per-token memory, cleaner multi-GPU pipelining, and allows smaller models to match the encrypted per-token latency of much larger transformer systems (1.5B at 79 s/token, 0.4B at 19 s/token; Table 15). Fully encrypted inference reaches 24 blocks at d=2048 with 4 CKKS bootstraps (correlation 0.99999; Table 17), and the combined pipeline matches plaintext exactly across 50 E2E queries on SQuAD and MS-MARCO (Table 14).

GPU-accelerated SIMD batching makes encrypted similarity search practical. FHE-Sim predicts accuracy without running encryption, allowing rapid configuration search.

Limitations. Fully encrypted BSGS with bootstrap reaches 24 blocks at d=2048 (correlation 0.99999) but currently covers only the FFN path (x^2 activation); the attention path requires $\sim$42 multiplicative levels per block (4 polynomial sigmoids at 5 levels each plus element-wise operations), and normalization layers require $1/\sqrt{\cdot}$ over a 5-order-of-magnitude variance range ($[6\times10^{-4}, 2\times10^{4}]$) that no polynomial can approximate, keeping both client-side. Bootstrap error ($\sim$0.025/refresh) accumulates linearly. **Broader impact.** Private RAG allows confidential document search and question answering where neither queries nor retrieved content are revealed to the server. FHE-native access control (Appendix C) provides per-user authorization with user anonymity via additive noise cancellation.

A Ranking Knowledge Distillation

This appendix presents ablations using weaker base embeddings (all-MiniLM-L6-v2, 384d) to isolate the contribution of projection techniques. With MRL-trained embeddings like Qwen3-Embedding, simple truncation achieves 92.3% R@10 at 64d, making learned projections less critical. However, for non-MRL embeddings, the techniques below remain relevant.

For high compression ratios (384d to 16d) on non-MRL embeddings, we compare loss functions and architectures.

A.1 Loss Function

InfoNCE preserves ranking but not similarity magnitudes. Under FHE noise, pairs with small similarity differences become indistinguishable. Ranking KD transfers the full similarity distribution from the original space:

$$\mathcal{L}_{\text{RKD}} = \text{KL}\left(\text{softmax}(s^{16}/\tau) \,||\, \text{softmax}(s^{384}/\tau)\right) \quad (17)$$

where s^{16} and s^{384} are pairwise similarities in projected and original space, $\tau = 0.1$.

A.2 Complex-Valued Projection

CKKS slots are complex numbers. A projection network with complex arithmetic matches this structure:

$$h_r = \text{GELU}(W_r x_r - W_i x_i) \quad (18)$$

$$h_i = \text{GELU}(W_r x_i + W_i x_r) \quad (19)$$

where x_r, x_i partition input dimensions. The output maps directly to complex packing. ComplexMLP has half the parameters of RealMLP (shared W_r, W_i across real/imaginary paths). The gain varies with training set size: +8% at 200 samples, +2% at 1600 samples.

Table 19: Projection ablation (384d $\rightarrow$ 16d, MS MARCO, R@10).

Configuration	R@10	Δ
InfoNCE + RealMLP	72.0%	—
Ranking KD + RealMLP	78.7%	+6.7
Ranking KD + ComplexMLP	79.5%	+7.5
Ranking KD + ComplexMLP + tuning	82.2%	+10.2

Code available at https://github.com/mozendr/FHE-SPEAR.

B Context-Dependent Generation Under FHE

Table 15 uses a factual prompt ("What is the capital of France?") where the correct answer is also in the model's training data. To confirm that encrypted generation depends on the passage, we test six prompts whose answers can only come from the

provided context: fictional entities, a counterfactual claim, a negation, and an acronym expansion. Each prompt is processed through the full 24-block 1.5B model (d=2048) with client-aided BSGS on an A100 (N=8,192, L_0=3). All FHE outputs match plaintext exactly (correlation 1.0).

Table 20: Context-dependent generation under FHE (A100, 1.5B). All outputs match plaintext exactly. The counterfactual overrides memorized knowledge (Paris becomes Lyon).

Query type	FHE output	s/tok
Counterfactual	"Lyon…" (5 tok)	259
Fictional entity	"NovaTech Industries' flagship" (5 tok)	252
Numerical fact	"The new catalyst works at" (5 tok)	250
Fictional expedition	"The expedition discovered a previously" (5 tok)	246
Negation	"The Aurora flower is **" (5 tok)	258
Acronym	"The acronym PRISM stands" (8 tok)	255

The counterfactual prompt asks "What is the capital of France?" with a passage stating Napoleon moved the capital to Lyon. Without context, RWKV-7 answers "Paris"; with the counterfactual passage under FHE, both the 1.5B and 0.4B models generate "Lyon." The 0.4B also extracts "The Helios" for the fictional NovaTech query (110 s/token). "NovaTech Industries" and the "Kepler Deep expedition" are fabricated and cannot be answered from training data. The Aurora flower prompt states the flower is crimson despite its name; the PRISM prompt defines a fictional acronym (Photonic Reconfigurable Integrated Sensor Matrix). In both cases the model extracts the passage-specific answer. Average latency: 253 s/token (1.5B), 111 s/token (0.4B).

B.1 Extended RAG Generation

To test sustained correctness, we generate 50 tokens from a RAG query over 100 passages (1 gold + 99 distractors) using the 1.5B configuration. Retrieval ranks the gold passage first:

> **Query:** "How did the COVID-19 pandemic affect the world?"
>
> **FHE output (50 tokens):** "The COVID-19 pandemic had a profound impact on the world, affecting every aspect of life. It caused widespread disruption to healthcare systems, economies, and daily life, leading to unprecedented challenges and changes. The pandemic resulted in over 7 million confirmed deaths globally"

The "7 million" figure comes from the retrieved passage rather than memorized knowledge. On privacy-sensitive data (replacing the gold passage with a synthetic patient record among 99 SQuAD distractors), the query "What treatment was recommended?" produces "…concurrent chemoradiation with carboplatin and paclitaxel…," extracting specific drug names from the encrypted record. Additional 50+ token runs across medical, historical, and encyclopedic domains all maintain exact plaintext match and produce coherent, grounded outputs.

C FHE-Native Access Control

Prior FHE-based RAG systems address encrypted retrieval [16], protect only embeddings with plaintext generation [17], or assume a separate FHE-compatible LLM for generation [15], but none demonstrate both encrypted retrieval and encrypted generation within a single pipeline. SecureRAG [15] employs lattice-based KP-ABE (via PALISADE-abe) as a separate cryptographic layer for access control. We achieve analogous per-user access control using only CKKS homomorphic addition, with no additional primitive, library, or multiplicative depth.

C.1 Protocol

Setup (admin, one-time). Passages are classified by PII content into sensitivity classes $\mathcal{C} = \{c_1, \ldots, c_K\}$ (e.g., financial, personal, temporal, organizational). For each class c, a random complex noise vector $\mathbf{n}_c \in \mathbb{C}^s$ is sampled with $\|\mathbf{n}_c\| = \alpha \cdot \overline{\|\mathbf{e}\|}$, where $\alpha \gg 1$ is the noise-to-signal ratio and s is the number of SIMD slots per document. Each restricted passage j of class c is stored as $\mathrm{Enc}(\mathbf{e}_j + \mathbf{n}_c)$; unrestricted passages are stored as $\mathrm{Enc}(\mathbf{e}_j)$. Passages with multiple classes (e.g., financial $\cap$ personal) receive the sum of class noises.

Correction generation (per-user). For user u with authorized classes $\mathcal{A}_u \subseteq \mathcal{C}$: for each class c and each SIMD batch b, the admin generates a correction ciphertext with $-\mathbf{n}_c$ in the slots occupied by class-c passages (if $c \in \mathcal{A}_u$) or random values $\mathbf{r} \sim \mathcal{CN}(0, \|\mathbf{n}_c\|^2)$ in those slots (if $c \notin \mathcal{A}_u$). Under CKKS IND-CPA security, the server cannot distinguish real corrections from dummy corrections.

Query (server-side). The server receives all $|\mathcal{C}|$ corrections per batch and applies them via `phantom.add`:

$$\mathrm{Enc}(\mathbf{e}_j + \mathbf{n}_c) + \mathrm{Enc}(-\mathbf{n}_c) = \mathrm{Enc}(\mathbf{e}_j) \tag{20}$$

For authorized classes, noise cancels exactly. For unauthorized classes, the dummy correction adds further random noise: $\mathrm{Enc}(\mathbf{e}_j + \mathbf{n}_c + \mathbf{r})$, which remains noise-dominated. Homomorphic addition consumes zero multiplicative levels, leaving the full modulus chain available for retrieval and generation.

C.2 PII Classification

On 100 MS-MARCO passages, PII detection identifies 44 sensitive spans across 22 passages (Table 21). Each span maps to one of four sensitivity classes.

C.3 Noise Security Analysis

Noise magnitude is controlled by the scale parameter $\alpha = \|\mathbf{n}_c\| / \overline{\|\mathbf{e}\|}$. Table 22 shows the signal-to-noise ratio for unauthorized retrieval scores as α varies.

Table 21: PII detection and sensitivity classification (MS-MARCO, 100 passages).

PII type	Count	Class
Monetary ($)	20	financial
Year reference	7	temporal
Phone number	5	personal
Percentage	5	financial
Date	3	temporal
Organization	3	organizational
Email	1	personal
Total	**44**	**4 classes**

Table 22: Noise security sweep: authorized vs. unauthorized similarity scores at different noise scales α. Separation is the ratio of unauthorized to authorized mean absolute score.

α	Auth $\bar{s}$	Auth σ	Unauth $\bar{s}$	Sep.
1	-0.26	0.11	0.18	1.5×
10	-0.26	0.11	4.38	17×
50	-0.26	0.11	23.0	90×
100	-0.26	0.11	46.3	181×
500	-0.26	0.11	232.9	908×

At $\alpha = 100$ (default), unauthorized scores are dominated by noise (181× separation), making correct retrieval impossible without the real correction.

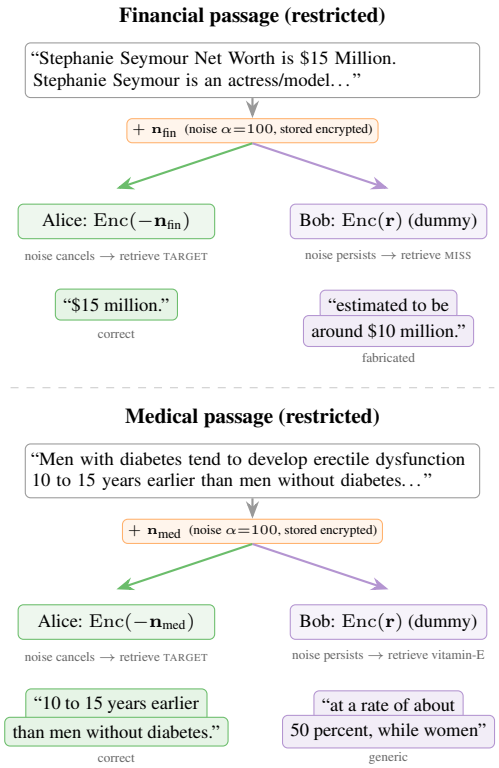

Figure 5: Access control demos (MS-MARCO, A100). Restricted passages stored with additive noise. Alice (authorized) retrieves the target and generates the exact statistic. Bob (unauthorized) retrieval misses; generation produces unrelated text.

C.4 Access Control Experiments

We test two users on the encrypted MS-MARCO corpus (100 passages, α=100, CT-CT retrieval with N=4096, generation with N=8192, L_0=3, 54-bit primes on A100):

- Alice (full access): correction ciphertexts cancel all class noise.
- Bob (temporal only): real corrections for temporal class, dummy for financial/personal/organizational/medical.

Table 23: FHE access control retrieval (A100, 100 MS-MARCO passages, α=100, CT-CT with N=4096).

	Baseline	Alice (full)	Bob (temporal)
R@1 (3 queries)	2/3	2/3	0/3
Correction time	–	0.10 ms	0.06 ms
Chain index	1	$1 \to 1$	$1 \to 1$
Corrections	–	5 real, 0 dummy	1 real, 4 dummy

Alice matches the baseline on all 3 queries; Bob's noise-dominated scores yield 0/3. Figure 5 shows two targeted demos with 11 tokens generated under FHE per user (1.5B RWKV-7, d=2048, 24 blocks, BSGS on A100). All 44 tokens match plaintext exactly (correlation 1.0).

C.5 Security Guarantees

Security properties:

1. Server blindness: the server never observes queries, scores, documents, or generated text.
2. Per-user access: different corrections yield different effective corpora.
3. User anonymity: real and dummy corrections are indistinguishable under CKKS IND-CPA.
4. Zero overhead: homomorphic addition only, zero multiplicative levels consumed.
5. Unified scheme: retrieval, access control, and generation all use the same CKKS parameters.

Revocation: the admin re-generates correction ciphertexts for affected classes. Passages need not be re-encrypted (noise is permanent); only per-user corrections change. Cost: $O(|\mathcal{C}| \cdot B)$ encryptions where B is the number of SIMD batches.

Limitations: noise is per-class, so score *differences* within a class are preserved ($\langle q, e_1{+}n_c\rangle - \langle q, e_2{+}n_c\rangle = \langle q, e_1{-}e_2\rangle$), leaking relative similarity. Multiple queries could cluster passages by class via correlated offsets. Per-passage noise eliminates this: each passage receives an independent random vector, using the same homomorphic addition and zero multiplicative levels, at $O(|\text{passages}|)$ vs $O(|\mathcal{C}|)$ corrections.

We verify experimentally on 100 MS-MARCO passages (10 queries, α=100). With per-class noise, within-class pairwise score-difference correlation is r=1.0 (ordering fully leaked). With per-passage noise, r=-0.07 (noise-to-signal ratio 1208×; ordering destroyed). Authorized retrieval with per-passage cor-

rections matches the no-noise baseline exactly (R@1 = 9/10), while unauthorized retrieval drops to 0/10. Collusion between an authorized user and the server breaks the scheme, as in most FHE-based access control.

D Prediction Formula Evaluation

We evaluate the prediction formula $\rho_{\text{FHE}} = \rho_{\text{compression}} \times \rho_{\text{noise}}$ against TenSEAL CKKS [5]. The test grid spans 11 embedding models (384d–1024d), 4 CKKS parameter sets ($N \in \{4096, 8192, 16384\}$, scale $\in \{2^{20}, 2^{40}\}$), and 9 target dimensions (8–256), yielding 396 configurations. Text data is 200 MS-MARCO passages with a 60/40 train/test split for SVD. For each configuration, we compare the predicted ρ_{FHE} against the actual Pearson correlation between original full-dimension similarities and encrypted compressed similarities (300 pairs).

Table 24: FHE-Sim prediction error by embedding model (4 CKKS configs $\times$ 9 target dims each).

Model	Dim	Mean	Max
all-mpnet-base-v2	768	0.66%	2.30%
Snowflake/arctic-embed-m	768	0.69%	2.08%
all-MiniLM-L6-v2	384	0.85%	3.26%
BAAI/bge-base-en-v1.5	768	1.65%	7.21%
multilingual-e5-large	1024	1.77%	9.32%
mxbai-embed-large-v1	1024	1.85%	8.30%
nomic-embed-text-v1.5	768	1.92%	7.31%
BAAI/bge-large-en-v1.5	1024	1.92%	8.40%
BAAI/bge-small-en-v1.5	384	2.14%	7.86%
intfloat/e5-small-v2	384	2.56%	10.81%
thenlper/gte-small	384	2.85%	13.32%
Overall		**1.72%**	**13.32%**

The median error is 1.13% (90th percentile: 3.63%). Models with higher intrinsic similarity variance (mpnet, Snowflake) are predicted more accurately because the signal-to-noise ratio is higher. The worst errors (10–13%) occur at high target dimensions with 20-bit scale ($c \approx 0.003$–0.005), where both $\rho_{\text{compression}}$ and ρ_{noise} are moderate and estimation errors in either factor compound. Furthermore, these errors only appear when both predicted and actual $\rho_{\text{FHE}} < 0.55$; for configurations where the prediction exceeds 0.9, the maximum error is 3%. At 40-bit scale, noise is negligible ($c \sim 10^{-8}$, $\rho_{\text{noise}} = 1.0$) and the prediction reduces to $\rho_{\text{compression}}$ alone (1.12% mean error). Overall bias is $+0.27\%$, indicating the noise independence assumption holds in practice.

References

[1] J.H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology – ASIACRYPT 2017*, LNCS vol. 10624, pp. 409–437. Springer, 2017.

[2] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proc. 33rd International Conference on Machine Learning (ICML)*, PMLR vol. 48, pp. 201–210, 2016.

[3] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *Proc. 27th USENIX Security Symposium*, pp. 1651–1668, 2018.

[4] T. Wang and P. Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proc. 37th International Conference on Machine Learning (ICML)*, PMLR vol. 119, pp. 9929–9939, 2020.

[5] A. Benaissa, B. Retiat, B. Cebere, and A.E. Belfedhal. TenSEAL: A library for encrypted tensor operations using homomorphic encryption. In *ICLR 2021 Workshop on Distributed and Private Machine Learning (DPML)*, 2021.

[6] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

[7] A. Kusupati, G. Bhatt, A. Rege, M. Wallingford, A. Sinha, V. Ramanujan, W. Howard-Snyder, K. Chen, S. Kakade, P. Jain, and A. Farhadi. Matryoshka representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 30233–30249, 2022.

[8] M. Nickel and D. Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[9] H. Yang, S. Shen, W. Dai, L. Zhou, Z. Liu, and Y. Zhao. Phantom: A CUDA-accelerated word-wise homomorphic encryption library. *IEEE Transactions on Dependable and Secure Computing*, 2024.

[10] B. Peng, R. Zhang, D. Goldstein, E. Alcaide, Q. Anthony, A. Albalak, S. Biderman, et al. RWKV-7 "Goose" with expressive dynamic state evolution. *arXiv preprint arXiv:2503.14456*, 2025.

[11] A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[12] H. Hou and J. Yang. EmbeddingRWKV: State-centric retrieval with reusable states. *arXiv preprint arXiv:2601.07861*, 2026.

[13] S. Halevi and V. Shoup. Algorithms in HElib. In *Advances in Cryptology – CRYPTO 2014*, LNCS vol. 8616, pp. 554–571. Springer, 2014.

[14] J.H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song. Bootstrapping for approximate homomorphic encryption. In *Advances in Cryptology – EUROCRYPT 2018*, LNCS vol. 10820, pp. 360–384. Springer, 2018.

[15] A. Bassit and V.N. Boddeti. SecureRAG: End-to-end secure retrieval-augmented generation. In *NeurIPS 2025 Workshop on GenAI for Health*, 2025.

[16] D. Zhao. FRAG: Toward federated vector database management for collaborative and secure retrieval-augmented generation. *arXiv preprint arXiv:2410.13272*, 2024.

[17] C. Qian, H. Zhang, Y. Tong, H.-W. Zheng, and Z. Zheng. HyFedRAG: A federated retrieval-augmented generation framework for heterogeneous and privacy-sensitive data. *arXiv preprint arXiv:2509.06444*, 2025.

[18] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma. CrypTFlow2: Practical 2-party secure inference. In *Proc. 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 325–342, 2020.

[19] Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.