Arion:
Attention-optimized transformer inference on encrypted data



Jingwei Chen
Chongqing Institute of Green and Intelligent Technology, CAS

November 6, 2026 @ Xi'an

L. Yang, J. Chen, W. Dai, S. Wang, W. Wu, Y. Feng. Expert Systems with Applications. doi: 10.1016/j.eswa.2026.134756

Roadmap

1 Introduction

- Background
- FHE
- Transformer

2 Arion

- Double BSGS
- Linear and nonlinear fusion
- Performance

Powerful assistants need private inputs

AI-assisted mathematics

The Exponential Time Hypothesis is False

OpenAI math preprint. (Oct. 25)

OpenAI also announced a Navier–Stokes result with smooth forcing. (Sept. 8)

Powerful assistants need private inputs

AI-assisted mathematics

The Exponential Time Hypothesis is False

OpenAI math preprint. (Oct. 25)

OpenAI also announced a Navier–Stokes result with smooth forcing. (Sept. 8)

Unpublished work and trust

Buckmaster raised concerns about drafts submitted to AI tools. (Approximately 12 hours before OpenAI's announcement)

OpenAI denied using those sessions in its competing work. (Sept. 10)

Powerful assistants need private inputs

AI-assisted mathematics

The Exponential Time Hypothesis is False

OpenAI math preprint. (Oct. 25)

OpenAI also announced a Navier–Stokes result with smooth forcing. (Sept. 8)

Unpublished work and trust

Buckmaster raised concerns about drafts submitted to AI tools. (Approximately 12 hours before OpenAI's announcement)

OpenAI denied using those sessions in its competing work. (Sept. 10)

Can a model process our input while the service provider cannot read it?

This motivates input confidentiality.

The task: evaluate a fixed map on a private matrix

$$f_\theta : \mathbb{R}^{m \times d} \longrightarrow \mathbb{R}^p,$$
$$\boldsymbol{X} \longmapsto f_\theta(\boldsymbol{X}).$$

The client holds $\boldsymbol{X}$

A private sequence represented by vectors.

m: sequence length. d: vector dimension.

The server holds θ

Model parameters learned previously.

Inference means evaluating this fixed map.

Goal: return an encrypted approximation to $f_\theta(\boldsymbol{X})$.

The output may be features or task scores used for classification.

Different technologies protect different things

Approach	Main idea	Main consideration
TEE	Isolated hardware execution	Trust in hardware and its implementation
DP	Limit information in released results	Privacy–utility tradeoff
MPC	Compute jointly across parties	Communication and interaction
FHE	Compute directly on ciphertexts	High computation cost

Arion targets non-interactive encrypted inference for LLMs with FHE.

TEE: Trusted execution environment; DP: Differential privacy; MPC: Secure multi-party computation; FHE: Fully homomorphic encryption.

Roadmap

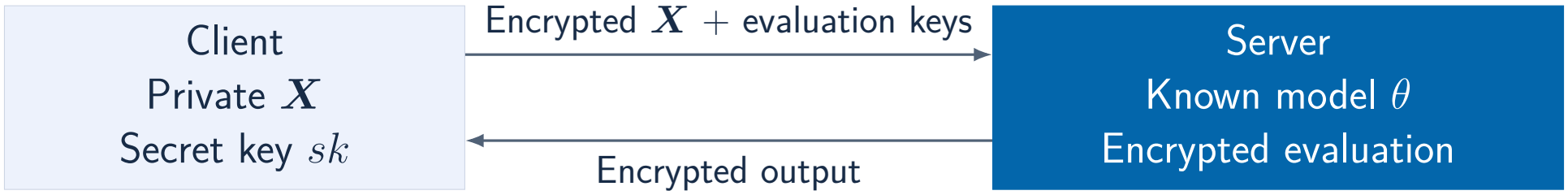
1 Introduction

- Background
- FHE
- Transformer

2 Arion

- Double BSGS
- Linear and nonlinear fusion
- Performance

Fully homomorphic encryption: the computation protocol



$$\mathrm{Dec}_{sk}(\mathrm{Eval}_{f_\theta}(\mathrm{Enc}(\boldsymbol{X}))) \approx f_\theta(\boldsymbol{X}).$$

The server computes without decrypting intermediate values.

Security model: the server follows the protocol but tries to infer private data.

Gentry, Craig. Fully homomorphic encryption using ideal lattices. in: STOC '09.

CKKS: encode a vector, then encrypt a polynomial

$$\underbrace{\boldsymbol{z} \in \mathbb{C}^{N/2}}_{\text{Message}} \xrightarrow{\text{Encode}_\Delta} \underbrace{\mu(X) \in R}_{\text{Plaintext}} \xrightarrow{\text{Enc}_{pk}} \underbrace{\boldsymbol{c} = (c_0, c_1) \in R_q^2}_{\text{Ciphertext}}.$$

$$R = \mathbb{Z}[X]/(X^N + 1), \quad R_q = R/qR, \quad N \text{ a power of two.}$$

One ciphertext, many coordinates

The vector coordinates are called *slots*.

Real vectors are a special case.

Approximate arithmetic

Δ controls the encoding scale.

Scaling, rounding and encryption introduce numerical error.

We will work with the vector operations induced on these slots.

Cheon, J.H., Kim, A., Kim, M., Song, Y. Homomorphic encryption for arithmetic of approximate numbers, in: ASIACRYPT '17.

SIMD arithmetic interface

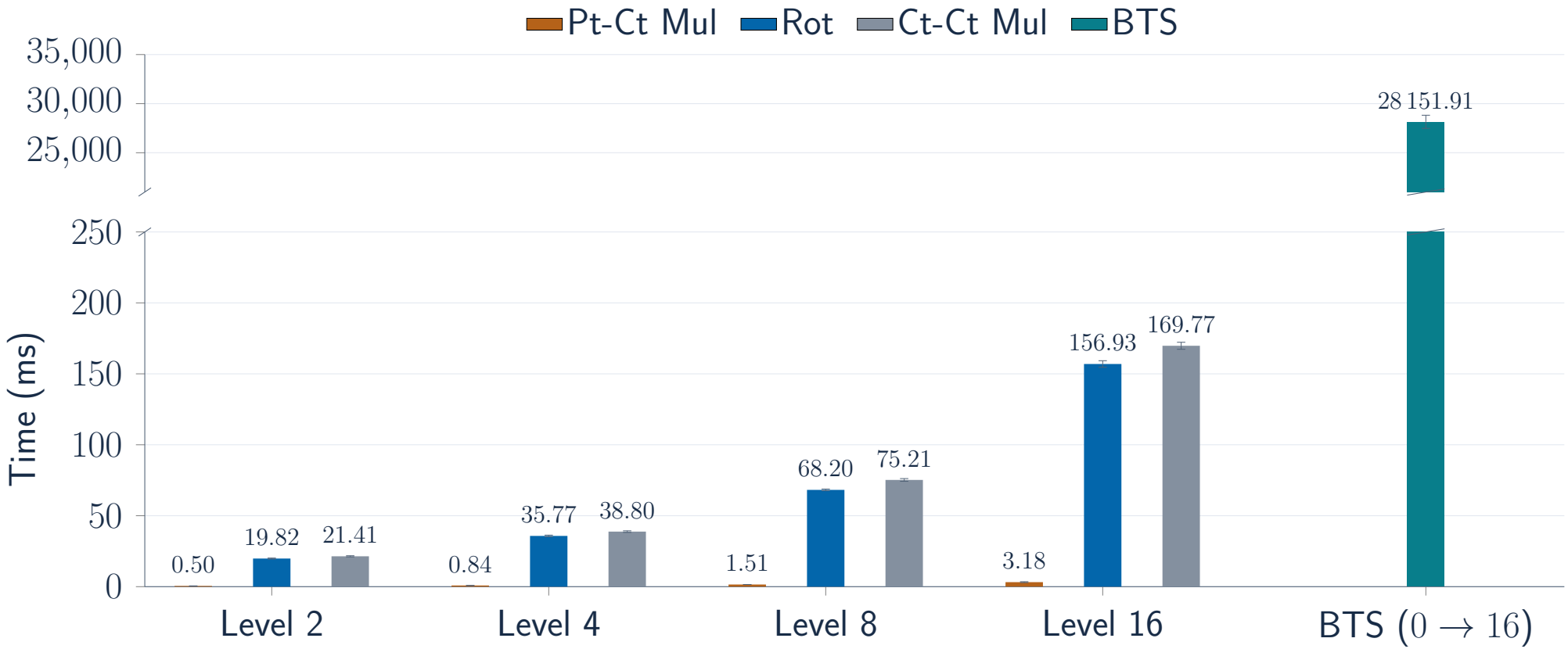
Write $[\![\boldsymbol{z}]\!]$ for an encryption of a packed vector $\boldsymbol{z}$.

Operation	Effect after decryption	Main cost consideration
Add	$\boldsymbol{x} + \boldsymbol{y}$	Componentwise arithmetic
Multiply	$\boldsymbol{x} \odot \boldsymbol{y}$	Consumes computation levels
Rotate	$\rho_r(\boldsymbol{x})_i = x_{(i+r) \bmod s}$	Expensive key switching

$\odot$: componentwise multiplication; $s = N/2$; All displayed effects are approximate.

A rotation that is cheap in plaintext can be expensive on ciphertexts.
Bootstrapping refreshes the usable level budget, very expensive.
Nonpolynomial operations, e.g., exponentials and reciprocals require approximation procedures.

The cost of basic homomorphic operations



Roadmap

1 Introduction

- Background
- FHE
- Transformer

2 Arion

- Double BSGS
- Linear and nonlinear fusion
- Performance

Embedding and positioning: from a sequence to a matrix



$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_0^\top \\ \boldsymbol{x}_1^\top \\ \vdots \\ \boldsymbol{x}_{m-1}^\top \end{pmatrix} \in \mathbb{R}^{m \times d}, \qquad \boldsymbol{x}_t \in \mathbb{R}^d.$$

The model updates a sequence of vectors using the other positions.

Attention is an input-dependent weighted average

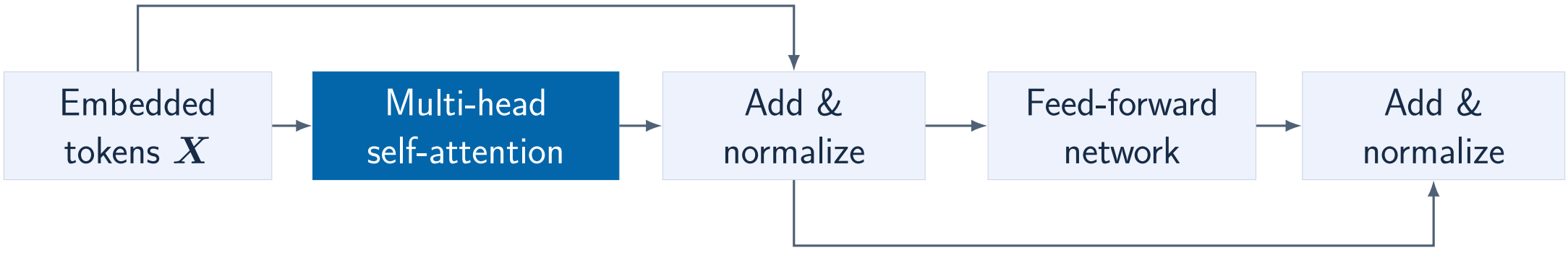
$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q, \quad \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K, \quad \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V \in \mathbb{R}^{m \times d_h},$$

$$\boldsymbol{A} = \operatorname{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_h}}\right)\boldsymbol{V}, \qquad \operatorname{softmax}(\boldsymbol{z}) = \left(\frac{e^{z_i}}{\Sigma_j\, e^{z_j}}\right).$$

$\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{d \times d_h}$; biases omitted; d_h is the dimension used by one attention head.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I. Attention is all you need, in: NIPS 17.

Where attention sits in a Transformer



One *head* uses one set of query, key and value projections.

H heads run related attention maps in parallel; L blocks are stacked.

An *encoder* processes the input sequence to produce features for a task.

Our full-model experiments use BERT encoders for classification.

Devlin, J., Chang, M., Lee, K., et al., 2019. BERT: Pre-training of deep bidirectional transformers for language understanding, in: NAACL '19.

Why encrypted attention is difficult

$$\underbrace{\boldsymbol{XW}}_{\text{CPMM}} \longrightarrow \underbrace{\boldsymbol{QK}^\top}_{\text{CCMM \& CMT}} \xrightarrow{\text{softmax}} \underbrace{\text{softmax}(\boldsymbol{QK}^\top)\boldsymbol{V}}_{\text{encrypted product}}$$

Data movement

CPMM, CCMM, and CMT.

Matrix products need many rotations.

Intermediate layouts must match.

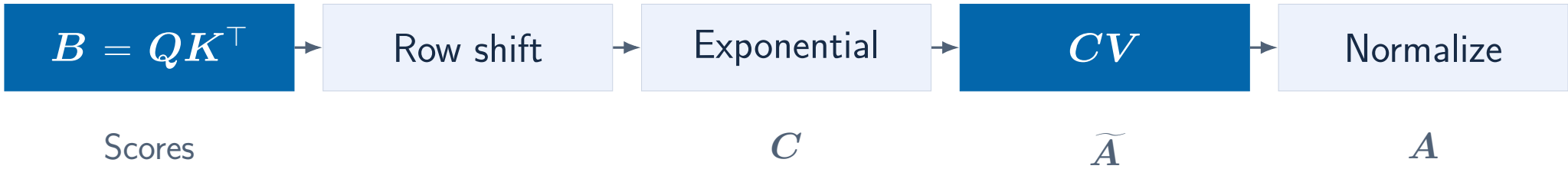
Nonlinear computation

Softmax needs shifts by max, exponentials and reciprocals.

Approximation must preserve accuracy.

$$\boldsymbol{A} = \text{softmax}\left(\frac{\boldsymbol{QK}^\top}{\sqrt{d_h}}\right)\boldsymbol{V}, \qquad \text{softmax}(\boldsymbol{z}) = \left(\frac{e^{z_i}}{\sum_j e^{z_j}}\right)$$

Our techniques: two complementary ideas



Double-BSGS

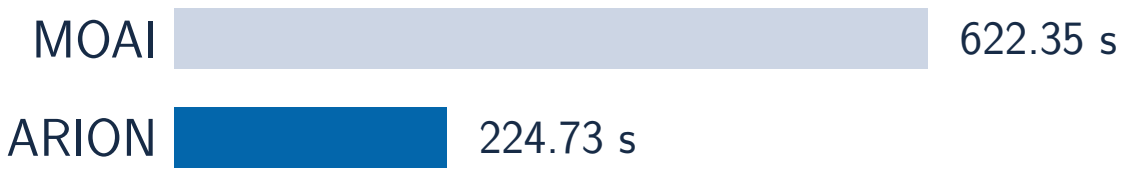
Reuse rotations in both encrypted matrix products.

Linear–nonlinear fusion

Carry a compatible representation through the softmax substeps.

$$A = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_h}}\right) V, \qquad \mathrm{softmax}(z) = \left(\frac{e^{z_i}}{\sum_j e^{z_j}}\right)$$

Our results: amortized BERT-Base inference time



2.77×
system speedup

$$\text{Reported seconds per input} = \frac{\text{batch runtime}}{256 \text{ inputs}}.$$

MOAI: SEAL, 112 threads. ARION: Lattigo, 64 threads.

Bootstrapping, depth and approximation choices also differ.

Batch size 256, sequence length 128; same workstation.

MOAI: Module-optimizing architecture for non-interactive secure transformer inference, ICLR '26, https://openreview.net/forum?id=qJn4HtTzhH

Roadmap

1 Introduction

- Background
- FHE
- Transformer

2 Arion

- Double BSGS
- Linear and nonlinear fusion
- Performance

Plain matrix–encrypted vector multiplication

$$\boldsymbol{A} = (a_{i,j})$$

a_{00}	a_{01}	a_{02}	a_{03}
a_{10}	a_{11}	a_{12}	a_{13}
a_{20}	a_{21}	a_{22}	a_{23}
a_{30}	a_{31}	a_{32}	a_{33}

Diagonal $r = 1$ wraps around

For $\boldsymbol{A} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{x} \in \mathbb{R}^m$:

$$\big(\boldsymbol{d}_r(\boldsymbol{A})\big)_i = a_{i,(i+r) \bmod m}, \qquad \rho_r(\boldsymbol{x})_i = x_{(i+r) \bmod m}.$$

For $m = 4$, $r = 1$:

$$\boldsymbol{d}_1(\boldsymbol{A}) = \begin{pmatrix} a_{01} \\ a_{12} \\ a_{23} \\ a_{30} \end{pmatrix}, \qquad \rho_1(\boldsymbol{x}) = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_0 \end{pmatrix}.$$

$$\boxed{\boldsymbol{A}\boldsymbol{x} = \sum_{r=0}^{m-1} \boldsymbol{d}_r(\boldsymbol{A}) \odot \rho_r(\boldsymbol{x})}$$

Pack each cyclic diagonal as a vector; multiply slotwise, then sum.

Online: $m - 1$ ciphertext rotations.

Leighton, F. T. (1992). Introduction to Parallel Algorithms and Architectures. San Mateo, Morgan Kaufmann Publishers.

Baby step giant step (BSGS) strategy

Giant step $4i$ \ Baby step j	0	1	2	3
	4	5	6	7
	8	9	10	11
	12	13	14	15

$$r = ik + j, \qquad m \leq k\ell$$

Example: $m = 16$, $k = \ell = 4$.

Reuse baby rotations of $\boldsymbol{x}$ and giant rotations of partial sums.

$k + \ell - 2 = 6$ nonzero shifts

$$\boxed{\boldsymbol{A}\boldsymbol{x} = \sum_{r=0}^{m-1} \boldsymbol{d}_r(\boldsymbol{A}) \odot \rho_r(\boldsymbol{x}) = \sum_{i=0}^{\ell-1} \rho_{ik}\left(\sum_{0 \leq j < k} \rho_{-ik}(\boldsymbol{d}_{ik+j}(\boldsymbol{A})) \odot \rho_j(\boldsymbol{x}) \right).}$$

Plaintext A: precompute and encode all $\rho_{-ik}(\boldsymbol{d}_{ik+j}(\boldsymbol{A}))$ offline.

Online: only $(k-1) + (\ell - 1)$ ciphertext rotations, with $k, \ell \approx \sqrt{m}$.

Halevi, S., Shoup, V., 2018. Faster homomorphic linear transformations in HElib, in: CRYPTO '18.

The obstacle: the intermediate matrix is private and dynamic

$$[\![\boldsymbol{Q}]\!],\ [\![\boldsymbol{K}]\!] \quad \longrightarrow \quad [\![\boldsymbol{B}]\!] \qquad \text{with} \qquad \boldsymbol{B} = \boldsymbol{Q}\boldsymbol{K}^{\top}.$$

For the remaining algebra, absorb the public factor $1/\sqrt{d_h}$ into $\boldsymbol{Q}$.

Fixed public matrix

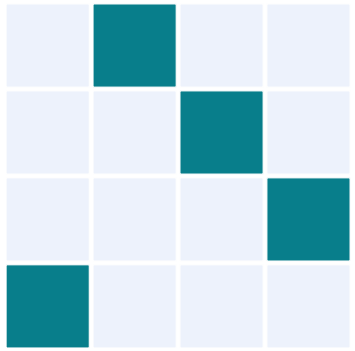
Rotated diagonals are known before the encrypted input arrives.

Attention score matrix

Its entries depend on the private input. Offline diagonal preparation is unavailable.

Arion constructs the required rotated diagonals directly from Q, K.

First BSGS: construct shifted score diagonals



A cyclic diagonal of $\boldsymbol{B}$

$\boldsymbol{B} = \boldsymbol{Q}\boldsymbol{K}^\top$, with $\boldsymbol{Q}, \boldsymbol{K} \in \mathbb{R}^{m \times d_h}$.

Store diagonals already rotated for the next product:

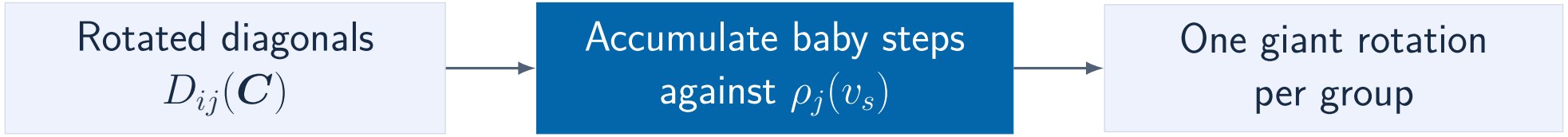
$$D_{ij}(\boldsymbol{B}) := \rho_{-ik}(d_{ik+j}(\boldsymbol{B}))\,.$$

Reuse rotated columns of $\boldsymbol{Q}$ and $\boldsymbol{K}$.

$$\boxed{D_{ij}(\boldsymbol{B}) = \sum_{s=0}^{d_h-1} \rho_{-ik}(\boldsymbol{q}_s) \odot \rho_j(\boldsymbol{k}_s)}$$

BSGS: $\boldsymbol{A}\boldsymbol{x} = \sum_{i=0}^{\ell-1} \rho_{ik}\Big(\sum_{0 \le j < k} \rho_{-ik}(\boldsymbol{d}_{ik+j}(\boldsymbol{A})) \odot \rho_j(\boldsymbol{x})\Big)$

Second BSGS: multiply by values using the same layout



$$(\boldsymbol{CV})_{:,s} = \sum_{i=0}^{\ell-1} \rho_{ik}\left(\sum_{\substack{0\le j<k \\ ik+j<m}} D_{ij}(\boldsymbol{C}) \odot \rho_j(v_s)\right)$$

$(k-1)$ baby rotations of v_s and $(\ell-1)$ giant rotations of partial sums.

Both products use at most $(k+\ell-2)d_h$ rotations each.

The rotation count gains a square-root factor

For H attention heads, each of width d_h:

$$O(md_hH) \quad \longrightarrow \quad O(\sqrt{m}\, d_hH).$$



82.5%

fewer rotations

BERT-Base: $m = 128$, $d_h = 64$, $H = 12$, $k = \ell = 12$.

This improves the rotation count; other arithmetic costs remain.

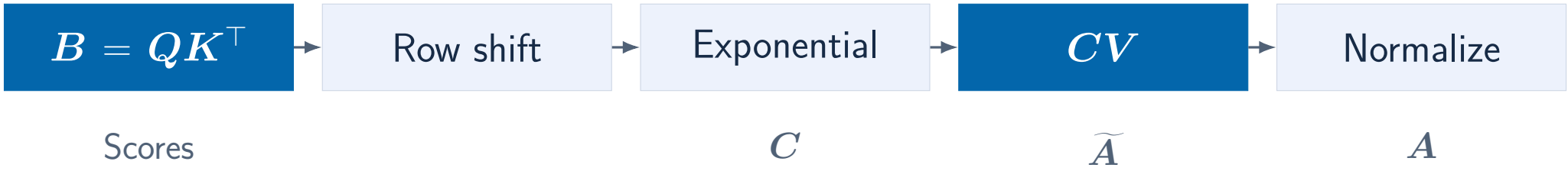
MOAI: Module-optimizing architecture for non-interactive secure transformer inference, ICLR '26, https://openreview.net/forum?id=qJn4HtTzhH

Roadmap

1 Introduction
- Background
- FHE
- Transformer

2 Arion
- Double BSGS
- Linear and nonlinear fusion
- Performance

The layout problem: row quantities must follow the rows



For $\boldsymbol{C} = (c_{i,j})$, let $s_i = \Sigma_j\, c_{i,j}$ and $r_i = 1/s_i$.

Rotated row order	2	3	0	1
Aligned reciprocals	r_2	r_3	r_0	r_1

$\rho_{-ik}(\boldsymbol{d}_{ik+j}(\boldsymbol{C}))$ must align with $\rho_{-ik}(\boldsymbol{r})$.

A *row shift* or *noramlization* must match the current coordinate rotation.

Fusion uses row-shift invariance and slotwise polynomials

A common shift cancels within each row

$$\frac{e^{b_{tu}-g_t}}{\sum_v e^{b_{tv}-g_t}} = \frac{e^{b_{tu}}}{\sum_v e^{b_{tv}}}.$$

Choose g_t from row statistics to control the numerical range.

Slotwise eval. commutes with rotation

$$p_{\exp}(\rho_r(\boldsymbol{x})) = \rho_r(p_{\exp}(\boldsymbol{x})).$$

Evaluate the polynomial directly on the rotated diagonals.

$$\boldsymbol{C} = \mathrm{Exp}_\circ(\boldsymbol{B} - \boldsymbol{g}\mathbf{1}^\top), \qquad \mathrm{Exp}_\circ \text{ means entrywise exponential.}$$

The nonlinear evaluation can preserve the rotated-diagonal layout.

Normalize after multiplying by values

$$\boldsymbol{C} \in \mathbb{R}^{m \times m}, \quad \boldsymbol{V} \in \mathbb{R}^{m \times d_h}, \quad \boldsymbol{s} = \boldsymbol{C}\mathbf{1},$$
$$\mathrm{softmax}(\boldsymbol{B})\boldsymbol{V} = \mathrm{diag}(\boldsymbol{s})^{-1}(\boldsymbol{C}\boldsymbol{V}).$$

Final row scaling	Before $\boldsymbol{CV}$	After $\boldsymbol{CV}$
Vectors to scale	m diagonals	d_h columns
Ciphertext multiplications	m	d_h
Reciprocal alignment	$\sqrt{m}$ rotations	No extra alignment

The product output already has the ordinary row order.

Inverses in the identity are approximated with the Goldschmidt algorithm.
BERT-Base: $m = 128$ and $d_h = 64$

Exact rearrangements, approximate evaluation

Exact algebra	Numerical approximation
Cyclic-diagonal identities	CKKS arithmetic
Permutation of slotwise products	Statistical row-shift estimation
Reassociation of row normalization	Exponential and reciprocal evaluation

The algebra reduces cost; task-level experiments assess retained accuracy.

Roadmap

1 Introduction
- Background
- FHE
- Transformer

2 Arion
- Double BSGS
- Linear and nonlinear fusion
- Performance

Controlled experiment: the matrix algorithm is faster

Model configuration	Without BSGS (s)	Double-BSGS (s)	Speedup
BERT-Tiny	365.5	196.8	**1.86**×
BERT-Base	2,239.3	1,184.4	**1.89**×
LLaMA-3-8B	10,051.2	5,694.1	**1.76**×

One layer's attention *matrix operations*, total time for 256 inputs.

Same code and hardware: 128 tokens per input, 64 threads.

The isolated matrix-operation gain is 1.76–1.89×.

Dual Xeon 6530 workstation, 1 TB RAM; Lattigo; same code base and hardware. LLaMA is a matrix-operation test only.

Devlin, J., Chang, M., Lee, K., et al., 2019. BERT: Pre-training of deep bidirectional transformers for language understanding, in: NAACL '19.
Grattafiori, A., Dubey, A., Jauhri, A., et al., 2024. The Llama 3 herd of models. https://llama.meta.com/

BERT-Base runtime, amortized per input

Module	Operations	MOAI	ARION
Attention	CPMM	39.46	13.35
	CCMM	51.13	24.28
	max		0.57
	exp	41.12[†]	3.29
	CCMM + bootstrapping + reciprocal		5.01
	Bootstrapping	106.04	32.89
Self Output	CPMM	1.83	5.20
	LayerNorm 1	0.48	0.43
	Bootstrapping	106.81	33.29
FFN	CPMM	51.61	19.80
	GELU	3.85	14.70
	CPMM	7.05	4.89
	Bootstrapping	106.41	33.49
Final	LayerNorm 2	0.49	0.41
	Bootstrapping	106.06	33.13
Total		**622.35**	**224.73**

[†] MOAI reports softmax and CCMM together.

Task accuracy stays close to plaintext inference

Model	Evaluation	SST-2	QNLI	RTE
		Sentiment	Question–sentence	Entailment
BERT-Tiny	Plaintext	81.31%	78.69%	60.29%
	Encrypted	81.42%	78.66%	59.93%
BERT-Base	Plaintext	92.66%	91.12%	70.40%
	Encrypted	92.09%	90.66%	69.31%

Classification accuracy on full development sets; higher is better.

The experiment decrypts the output features before the final classification step.

What this talk leaves out

Representation and scheduling

Interleaved batching

Packing choices across model stages

Bootstrapping placement

Numerical engineering

Polynomial approximation design

GELU and layer normalization

Precision management

These steps are part of the Arion system and described in the manuscript.

Conclusions and next steps

Our techniques

Construct reusable rotated diagonals from factored encrypted inputs.

Preserve their layout through nonlinear evaluation and normalize later.

Measured evidence

82.5% fewer rotations

1.76–1.89× matrix-operation speedup

2.77× system speedup on CPU

Next implementation targets

Full encrypted text generation: causal masks, grouped-query attention and key/value cache

GPU acceleration

The rotation ledger

Stage	Upper bound per head
Score product	$(k+\ell-2)d_h$
Statistical row shift	4ℓ
Slotwise exponential	0
Value product	$(k+\ell-2)d_h$
Normalization	$\ell-1$

$$\text{Total} \le H\big[2(k+\ell-2)d_h + 5\ell - 1\big].$$

The paper's simpler conservative bound gives $37{,}584$ rotations for $m = 128$, $d_h = 64$, $H = 12$, $k = \ell = 12$.

The reference MOAI count in the same conservative comparison is 215,040.

CKKS parameters and security scope

Reported parameters

$N = 2^{16}$, $\log_2 \Delta \approx 45$

$\log_2 Q = 58 + 45 \cdot 14$

$\log_2 P = 60 \cdot 4$

Bootstrap $\log_2(QP) = 1749$

Reported 128-bit security estimate

Guarantee and assumptions

Client-input confidentiality against a semi-honest server.

The secret key stays with the client.

Malicious security, circuit privacy and model privacy require additional analysis.

Workstation: two Xeon 6530 CPUs, 64 cores total, 1 TB RAM.

Token length and a real world application

BERT-Tiny

Max tokens	Batch	s/input
16	2,048	1.06
32	1,024	2.21
64	512	4.46
128	256	9.30

DASHformer on iDASH24

163 protein sequences, length 50

165 s total; 1.01 s/input
29 GB RAM; no bootstrapping

Reported micro-AUC: 0.99